NCS-TIB-88-6

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN 88-6

INVESTIGATION OF VECTOR QUANTIZATION FOR THE CODING OF GRAY SCALE IMAGES FOR GROUP 4 FACSIMILE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified

1b. RESTRICTIVE MARKINGS:

2a. SECURITY CLASSIFICATION AUTHORITY:

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE:

3. DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S): NCS TIB 88-6

5. MONITORING ORGANIZATION REPORT NUMBER(S):

6a. NAME OF PERFORMING ORGANIZATION: Delta Information Systems, Inc.

6b. OFFICE SYMBOL (If applicable):

7a. NAME OF MONITORING ORGANIZATION:

6c. ADDRESS (City, State, and ZIP Code): Horsham Business Center, Bldg. 3
300 Welsh Road
Horsham, PA 19044

7b. ADDRESS (City, State, and ZIP Code):

8a. NAME OF FUNDING / SPONSORING ORGANIZATION: National Communications System

8b. OFFICE SYMBOL (If applicable): NCS-TS

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: DCA100-87-C-0078

8c. ADDRESS (City, State, and ZIP Code): Office of Technology & Standards
Washington, D.C. 20305-2010

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
33127K	Q011	87-003	

11. TITLE (Include Security Classification): Investigation of Vector Quantization for the Coding of Gray Scale Images for Group 4 Facsimile

12. PERSONAL AUTHOR(S):

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	August 1988	100

16. SUPPLEMENTARY NOTATION:

17 COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Gray Scale Images Group 4 Facsimile

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Vector Quantization is a new gray scale coding technique showing the promise of large compression ratios and good picture quality. No comprehensive study analyzing Vector Quantization, as applied to Group 4 facsimile systems under carefully controlled conditions, has been performed prior to this investigation. The purpose of this study was to evaluate Vector Quantization to determine its relative effectiveness for the addition of gray scale to Group 4 facsimile.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION: Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL: Janet Orndorff

22b. TELEPHONE (Include Area Code): 202-692-2124

22c. OFFICE SYMBOL: NCS-TS

DD Form 1473, JUN 86 — Previous editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

NCS TECHNICAL INFORMATION BULLETIN 88-6

INVESTIGATION OF VECTOR QUANTIZATION FOR THE CODING OF GRAY SCALE IMAGES FOR GROUP 4 FACSIMILE

PROJECT OFFICER

Dennis Bodson

DENNIS BODSON
Assistant Manager
Office of NCS Technology
and Standards

APPROVED FOR PUBLICATION:

Dennis Bodson

DENNIS BODSON
Assistant Manager
Office of NCS Technology
and Standards

FOREWORD

Among the responsibilities assigned to the Office of the Manager, National Communications System, is the management of the Federal Telecommunication Standards Program. Under this program, the NCS, with the assistance of the Federal Telecommunication Standards Committee identified, develops, and coordinates proposed Federal Standards which either contribute to the interoperability of functionally similar Federal telecommunication systems or to the achievement of a compatible and efficient interface between computer and telecommunication systems. In developing and coordinating these standards, a considerable amount of effort is expended in initiating and pursuing joint standards development efforts with appropriate technical committees of the International Organization for Standardization, and the International Telegraph and Telephone Consultative Committee of the International Telecommunication Union. This Technical Information Bulletin presents and overview of an effort which is contributing to the development of compatible Federal, national, and international standards in the area of facsimile. It has been prepared to inform interested Federal activities of the progress of these efforts. Any comments, inputs or statements of requirements which could assist in the advancement of this work are welcome and should be addressed to:

Office of the Manager
National Communications System
ATTN: NCS-TS
Washington, DC 20305-2010

INVESTIGATION OF VECTOR QUANTIZATION FOR THE CODING OF GRAY SCALE IMAGES FOR GROUP 4 FACSIMILE

AUGUST, 1988

SUBMITTED TO:

NATIONAL COMMUNICATIONS AGENCY

Office of Technology and Standards

WASHINGTON, D.C. 20305

Contracting Agency:

DEFENSE COMMUNICATIONS AGENCY

Contract Number - DCA100-87-C-0078

Task Number - 87-003

D E L T A I N F O R M A T I O N S Y S T E M S, I N C
Horsham Business Center, Bldg. 3
300 Welsh Road
Horsham, PA 19044
TEL: (215) 657-5270 FAX: (215) 657-5273

INVESTIGATION OF VECTOR QUANTIZATION FOR THE CODING OF GRAY SCALE IMAGES FOR GROUP 4 FACSIMILE

TABLE OF CONTENTS

1.0 Introduction

This document summarizes work performed by Delta Information Systems, Inc., (DIS) for the Office of Technology and Standards of the National Communications System, an organization of the U. S. Government, headed by acting National Communications System Assistant Manager Dennis Bodson. Mr. Bodson is responsible for the management of the Federal Telecommunications Standards Program, which develops telecommunications standards, the use of which is mandatory for all Federal agencies. The purpose of this study, performed under task order number 87-003 of contract number DCA100-87-C-0078, was to investigate the effectiveness of Vector Quantization when applied to the transmission of gray scale imagery via Group 4 facsimile.

At the present time, there are no CCITT Recommendations that provide for the transmission of gray scale imagery via Group 4 facsimile; however, the CCITT is considering gray scale transmission as a Group 4 option.

Vector Quantization is a new gray scale coding technique showing the promise of large compression ratios and good picture quality. No comprehensive study analyzing Vector Quantization, as applied to Group 4 facsimile systems under carefully controlled conditions, has been performed prior to this investigation. The purpose of this study was to evaluate Vector Quantization to determine its relative effectiveness for the addition of gray scale to Group 4 facsimile.

This report is comprised of six sections. Section 1.0 provides a brief synopsis of the study objectives and outlines its results and conclusions. Section 2.0 describes the initial investigation, includes a highly simplified description of the Vector Quantization concept, and concludes with a brief outline of the selected approach and the reasons for its selection. Section 3.0 gives a system overview. Section 4.0 describes the system in detail. Section 5.0 presents the simulation parameters and results. Section 6.0 closes the report with conclusions and recommendations for future investigation.

1.1 Synopsis

The investigation was conducted in four major phases. The first phase consisted of a study of published Vector Quantization techniques and culminated in the selection of one to be evaluated during the balance of the investigation. Simulation software for the selected technique was developed in the second phase. The third phase was devoted to the simulation of the transmission and reception of the Standard Gray Scale Images,[1] and the fourth phase consisted of evaluating the results, writing the final report and preparing all deliverable items for delivery.

The simulated algorithm begins by dividing the starting image into equal-sized rectangular blocks of pixels. For each block, the transmitter computes and transmits the mean gray level value of the block pixels. It then compares the difference vector with a locally stored codebook (library) of normalized

(unit magnitude) vectors and selects that codebook vector which is best correlated with the difference vector. It transmits the dot product of the difference and codebook vectors, called "gain," and the index to the selected codebook vector. The receiver reconstructs an approximation of the starting image block from the transmitted data and a locally stored copy of the codebook.

This algorithm was employed to compress the standard gray scale images, which were originally digitized to 8 bits by pulse code modulation (PCM).

The present system achieves a compression rate of 1.3 - 1.5 bits per pixel with very good image quality and 1.1 - 1.2 bits with moderately good image quality. Of these 1.1 - 1.5 bits, one whole bit is currently required for the mean and gain values combined, and only 0.1 - 0.5 for the codebook vector index. It is estimated that the appropriate combination of differential pulse code modulation (DPCM), tailored to the mean and gain data statistics, and variable-length coding can compress the mean and gain data to approximately 0.5 bits per pixel. This would yield an overall compression rate of less than one bit per pixel.

2.0 Initial Investigation

2.1. Definition of Vector Quantization

2.1.1. Simplified Description

Vector Quantization beings by dividing an image to be transmitted into rectangular blocks of pixels, all blocks having the same dimensions. The transmitter compares each block with a large library of typical blocks, called a "codebook," and selects the library block that best approximates the block to be transmitted. The transmitter then encodes and transmits the index to the selected library block. The receiver, equipped with a copy of the codebook, decodes the index, retrieves the selected library block and inserts it into the output image.

This process is called Vector Quantization because, both theoretically and computationally, each block is treated as a vector. The vector representation of a block can be thought of as laying out all the gray-scale values of the block pixels in a single string, that of the upper left pixel first, and of the lower right last. Such a string of numbers comprises a vector in k-dimensional space, where k is the number of pixels in the block. When the block is treated in this manner, the entire body of mathematical knowledge of vector analysis and multi-dimensional analytical geometry can be brought to bear on the Vector Quantization problem. In the balance of this report, the terms "block" and "vector" will be used interchangeably, with "block" referring to a rectangular array of pixels in an image, and "vector" referring to the representation of these pixels as a

string of numbers.

In all the variations of Vector Quantization to be discussed below, there is a trade-off between image quality and data compression. In the theoretical limit of zero distortion, the codebook would contain vectors representing all possible blocks. An exact match would always be found. Distortionless transmission would, however, entail an enormous codebook and little data compression, even with optimal coding. At the other extreme, a codebook containing few vectors (representative blocks) would yield large compression ratios, but poor image quality. The objective of any Vector Quantization system design is, therefore, to achieve the best compromise among codebook size, data compression and received image quality.

2.1.2. Definition of Terms

The following terms apply generally to Vector Quantization; additional terms that apply specifically to the selected approach are defined later.

Vector

A vector is an ordered sequence of numbers, for example, the sequence of gray-scale values in one image block.

Vector Component

A vector component is one of the numbers in the sequence of numbers comprising a vector (for example, the gray-scale value of one pixel in a block).

Vector Magnitude

The magnitude of a vector is the square root of the sum of the squares of the vector components.

Distance Between Two Vectors

The distance between two vectors is the square root of the sum of the squares of the differences between the respective components of the two vectors. The distance between two identical vectors is zero.

Vector Distortion

Vector distortion refers to the distortion incurred in a block resulting from selecting a codebook vector that approximates the vector to be transmitted. In most Vector Quantization analyses, the distortion is defined as the distance between the two vectors, or the square of the distance. This definition is assumed throughout the balance of this report unless otherwise specified.

Centroid

The centroid of a set of vectors is a vector, each of whose components is the average of the corresponding components of all the vectors in the set. The centroid vector is not necessarily one of the vectors from which it is formed. The vector having minimum mean square distance to the vectors in the set is the centroid vector.

2.2. Methods Investigated

2.2.1. Literature Search

A review of published papers revealed many variations on the Vector Quantization theme. Gersho[2] presents a mathematical treatment of the problem. The codebook is, in effect, the vector quantizer in that it "quantizes" the multi-dimensional vector "space" into a finite set of representative vectors. Gersho goes on to explore the partitioning problem, and concludes that the only practical way to design the quantizer (select the vectors to be included in the codebook) is to take advantage of vector clustering.

The basic vector clustering algorithm was thoroughly developed by Linde, Buzo and Gray.[3] This algorithm, known as the LBG algorithm, takes advantage of the fact that the vector representations of image blocks tend to cluster in the vector space. A codebook containing vectors representing the cluster centroids offers the best compromise between codebook size and received image quality. The method consists of using a long sequence of "training" vectors to design the codebook. The authors note that this basic method yields a locally optimized codebook in that a slight perturbation of any codebook vector increases the average distortion with respect to the training vectors. The authors offer an interesting variation in which global optimization (best possible codebook for the training vector set) is more likely when the optimization is started with a large amount of random noise superimposed on the training

vectors. The optimization is refined by repeated iterations with the noise gradually reduced to zero.

Gray and Linde[4] explain and compare several variations on the LBG codebook generation method and compare the resulting performances of Gauss-Markov sources. In particular, the authors show that tree-assisted codebook searches allow the use of codebooks much larger than those practical with exhaustive searches at the expense of a suboptimal codebook. The performance is only slightly degraded with respect to the exhaustive-search approach.

Hang and Woods[5] discuss predictive vector quantization, which consists of a combination of predictive filtering and vector quantization. The purpose of the predictive filtering is to remove redundancy before vector quantizing the residue.

A vector quantization method that offers great promise of good compression and low distortion is described in Japan Annex 4.[6] This method combines DPCM (Differential Pulse Code Modulation) and vector quantization. Since this approach was selected for simulation, further description is deferred to a later section of this report.

Other references studied include Helden and Boekee,[7] Dutch PTT,[8] and Gersho and Ramamarthi.[9]

2.2.2. Common Underlying Issues

Vector Quantization, in all its forms, requires a large codebook of vectors from which one is selected for each block to

be transmitted. Two very important issues are therefore: (1) codebook search and (2) codebook generation.

2.2.2.1. Codebook Search

There are two basic search methods: exhaustive and tree-assisted. The exhaustive method is guaranteed to select the codebook vector that best matches the input vector (block in the input image). This method is practical, however, only for very small block sizes, because the search time grows exponentially with block size. A tree search is much faster. A binary tree search begins with a choice between two codebook vectors that act as "keys" to the next search level. The selection of one of these "keys" leads to another two-way choice, which leads to a better approximation of the input vector, which leads to yet another two-way choice, etc.. This method, though much faster than the exhaustive search, may fail to find the best match, because once a two-way choice has been made in a given tree level, the search may be directed to a subtree that does not contain the best match. The general m-ary tree search, in which an m-way choice is made at each decision level, gives better performance as the value of m increases, at the expense of longer search time. The exhaustive search is the limiting case of one M-way decision, where M is the total number of codebook vectors.

2.2.2.2. Codebook Generation

The codebook generation objective is a codebook that gives low image distortion while minimizing the codebook size. Minimizing the codebook size is important, not only to minimize memory and search time, but also to achieve high compression ratios.

All codebook generation methods reported in the literature are variations on the LBG (Linde, Buzo and Gray) method. In principle, if one knew the statistics of all images to be transmitted, one could generate a codebook analytically. The most commonly used method consists, however, of using a large number of training vectors, each training vector representing a "typical" image block. The use of training vectors is assumed in the balance of this report.

The literature makes frequent reference to training vectors, but gives little detail as to how they are derived. A common problem is a tendency to generate a codebook that performs well with the images from which the training vectors are derived, and poorly with others.

DIS devoted considerable effort to training vector generation and processing, as described in Sections 3.2, 4.2 and 5.1.2. The resulting codebooks were very robust, performing as well with images not included in the training vector set as those from which the training vectors were derived.

The following is a summary of the LBG codebook generation method. Assume, for the moment, a partially optimized codebook.

Each training vector "belongs" to a codebook vector in that the training vector matches the codebook vector at least as well as it matches any other. (Ties are broken in various ways depending on the specific method used.)

The codebook is updated to make each codebook vector the centroid of the set of training vectors that belong to it, thus minimizing the average distortion with respect to that set of training vectors. The update may, however, cause some of the training vectors that belonged to a given codebook vector before the update to belong to a different codebook vector afterward. Another iteration is therefore performed to compute new centroids, and the codebook is updated again. This process is repeated until there is no further improvement, or the improvement is less than some specified value.

This iterative method of codebook improvement leads to a local minimum of average distortion. A slight perturbation of the codebook vectors gives greater distortion. This method leaves the possibility that some large change to the codebook might give even less distortion; hence the local minimum is not necessarily the global minimum (best possible codebook for the training vector set).

Codebook generation begins with one codebook vector which is the centroid of all the training vectors. This vector is then split into two vectors very close to each other. The splitting objective is to make the numbers of training vectors belonging to the two codebook vectors approximately equal. The codebook is

then optimized, as described above. The two (now optimized) vectors are then split into four, and optimization is repeated. The process is continued until a codebook of the required size is achieved.

The literature suggests guidelines for the selection of the codebook size for various block sizes and distortion goals. Very little is said about how to split existing vectors. Both of these issues were addressed in depth during software design and development.

2.3. Selected Method

The selected method is based on the Japan Annex 4 paper, "Component Vector Quantization," Annex 4 of CCITT Study Group VIII, Geneva, 1-12 December 1986. This method will henceforth be referred to as the "selected method" or the "Japan Annex 4 method."

2.3.1. Simplified Description

For each block to be transmitted, the transmitter computes the mean (average) value of the vector components (block pixel gray levels) and subtracts this value from each component, leaving a zero-mean difference vector. The mean value is transmitted by DPCM (Differential Pulse Code Modulation). Next, the transmitter performs a binary tree search of a codebook whose vectors are normalized (see definition in Section 3.1). In essence, a normalized vector is one in which the information

describing the overall block brightness and contrast has been removed. The selected codebook vector is that which is most closely correlated with the difference vector. The correlation value, or "gain" (contrast information), is transmitted by DPCM. The index to the selected codebook vector is also transmitted.

The receiver retrieves the normalized codebook vector indicated by the codebook vector index, multiplies it by the "gain" (correlation value) and adds the mean to give an output vector which approximates the input vector.

The binary tree search is amplitude adaptive. A "flat" image block (all pixels having the same gray-scale value) can be described by the mean value alone, and no vector index need be transmitted. A nearly flat block requires only a coarse correlation to the normalized codebook vector because the block's contrast is so low that an accurate match is not required to give low distortion. The binary tree search is therefore terminated at low tree levels for low contrast blocks and at higher tree levels for higher contrast blocks. The codebook vector index codeword length is proportional to the block's contrast. Because low contrast blocks occur more frequently than high, short codebook vector index codes occur more frequently than long. This method is therefore very efficient.

2.3.2. Reason for Selection

The Japan Annex 4 method was selected because it is the most aggressive approach studied in terms of compression for a given

distortion. The codebook size is less for a given block size than with other methods, and the coding is more efficient.

The codebook size is smaller because the system removes a great deal of redundancy from the codebook vectors by removing mean and gain (brightness and contrast) data. Because of the relatively small codebook size for a given block size, a larger block is feasible than would be with other methods. The larger block spreads the "cost" of transmitting the mean and gain over more pixels.

Low contrast blocks require less accurate matching to a codebook vector than do high-contrast blocks. This can easily be seen by considering the limiting case of no contrast. In this case no vector index need be transmitted; the block information is contained entirely in the mean value. The codebook and accompanying search tree are designed to terminate the search at low tree levels for low contrast blocks. The codebook vector index codeword is short for low tree levels, longer for higher tree levels. Thus, the codeword length is proportional to the block contrast. Since low contrast blocks occur more frequently than high, short codes occur more frequently than long.

3.0 System Overview

This section gives an overview of the DIS implementation of the Japan Annex 4 Vector Quantization method. The next section describes the system in more detail.

3.1. Definition of Terms

The following terms are used to describe the DIS implementation.

Vector Mean

A vector's mean is the average value of the vector components.

Zero-Mean Vector

A zero-mean vector is a vector whose mean is zero. It is derived from a non-zero-mean vector by subtracting the vector mean from each vector component.

Normalized Vector

A normalized vector is a vector having unit magnitude. An unnormalized vector is normalized by computing the vector magnitude and dividing it into each component of the unnormalized vector. A vector whose components are all zero cannot be normalized, since the normalization process would entail dividing zero by zero. In the Japan Annex 4 context, normalized vectors are also zero-mean.

Dot Product

The dot product, also called inner product or scaler product, of two vectors is the sum of the products of their corresponding components. It is equal to the magnitude of one vector times the magnitude of the other times the cosine of the angle defined by the two vectors, with the angle vertex at the vector space origin.

Gain

The word "gain," as used in this report, is defined as the dot product of a difference (zero-mean) vector with the selected (normalized) codebook vector. The Japan Annex 4 paper defines gain (factor) as the magnitude of the difference vector itself, and calls the dot product the amplitude deviation. This terminology is avoided in this report; "amplitude deviation" could connote some kind of error, or deviation from some norm, and is therefore confusing. Moreover, the symbol for "amplitude deviation" in the reference is g. The words "magnitude" and "gain" in this document are therefore respectively equivalent to "gain factor" and "amplitude deviation" in the reference.

Search Tree

The codebook search tree is a binary tree structure with 1 node (decision point) at tree level 0 (the tree root), 2 at level 1, 4 at level 2, ..., 2^L at any given level, L. One codebook vector is assigned to each tree node.

Absolute Codebook Vector Index

The absolute index of a codebook vector is that vector's position in the codebook. The lowest index is 0.

Base Codebook Vector Index

For any given search tree level, the base codebook vector index is the lowest absolute index of the codebook vectors assigned to the tree nodes in that level. The codebook contains a list of base codebook vector indices for the various tree levels.

Relative Codebook Vector Index

The relative index of a codebook vector is that vector's position in the codebook relative to the base index for the vector's tree level, L. The lowest value is 0; the highest $2^L - 1$. A relative codebook vector index is therefore an L-bit binary number. This property is a system design cornerstone.

Gain Threshold

The gain threshold of a tree level controls the transmitter codebook search and the receiver's codebook lookup. The tree level of the selected codebook vector is such that the gain of the difference vector is at or below that level's threshold and above the threshold of all lower tree levels. The codebook contains a list of gain thresholds for the various tree levels.

3.2. System Description

The Vector Quantization system processes digital images only. Starting images are digitized to 8 bits per pixel, giving a gray scale of 0 (black) through 255 (white). The starting digital images are virtually indistinguishable from high-quality analog images. These starting images are the standards with which the ending images are compared.

Figure 3.1 shows the codebook generation subsystem. A training image (composite of actual images) is processed by a training vector generation program, which creates a file of normalized training vectors. The magnitudes of the zero-mean vectors prior to normalization are carried with the vectors.

Next, the training vectors are sorted into descending order of vector magnitude. This reduces the execution time of the codebook generator, as is explained in Section 4.2.4.

The codebook generator is based on the LBG algorithm. It automatically determines the tree level gain thresholds, which control the amplitude-adaptive tree searches. The thresholds are determined by a pre-specified signal-to-noise ratio goal. This automatic threshold generation feature may be novel; the Japan Annex 4 reference does not explain how the gain range for a given codebook vector codeword length is determined.

Figure 3.2 shows the transmitter simulation subsystem. The transmission simulation was implemented in two stages to employ a DPCM data compression system developed under another contract.[10]

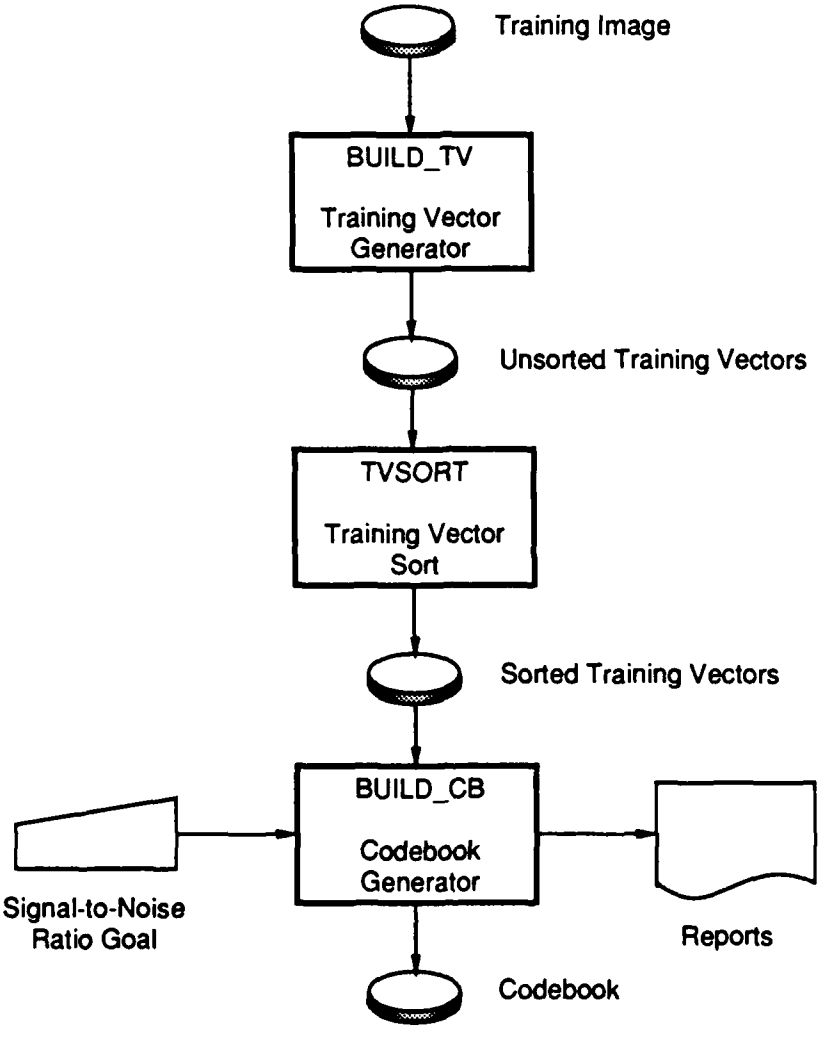


Figure 3.1 Codebook Generation

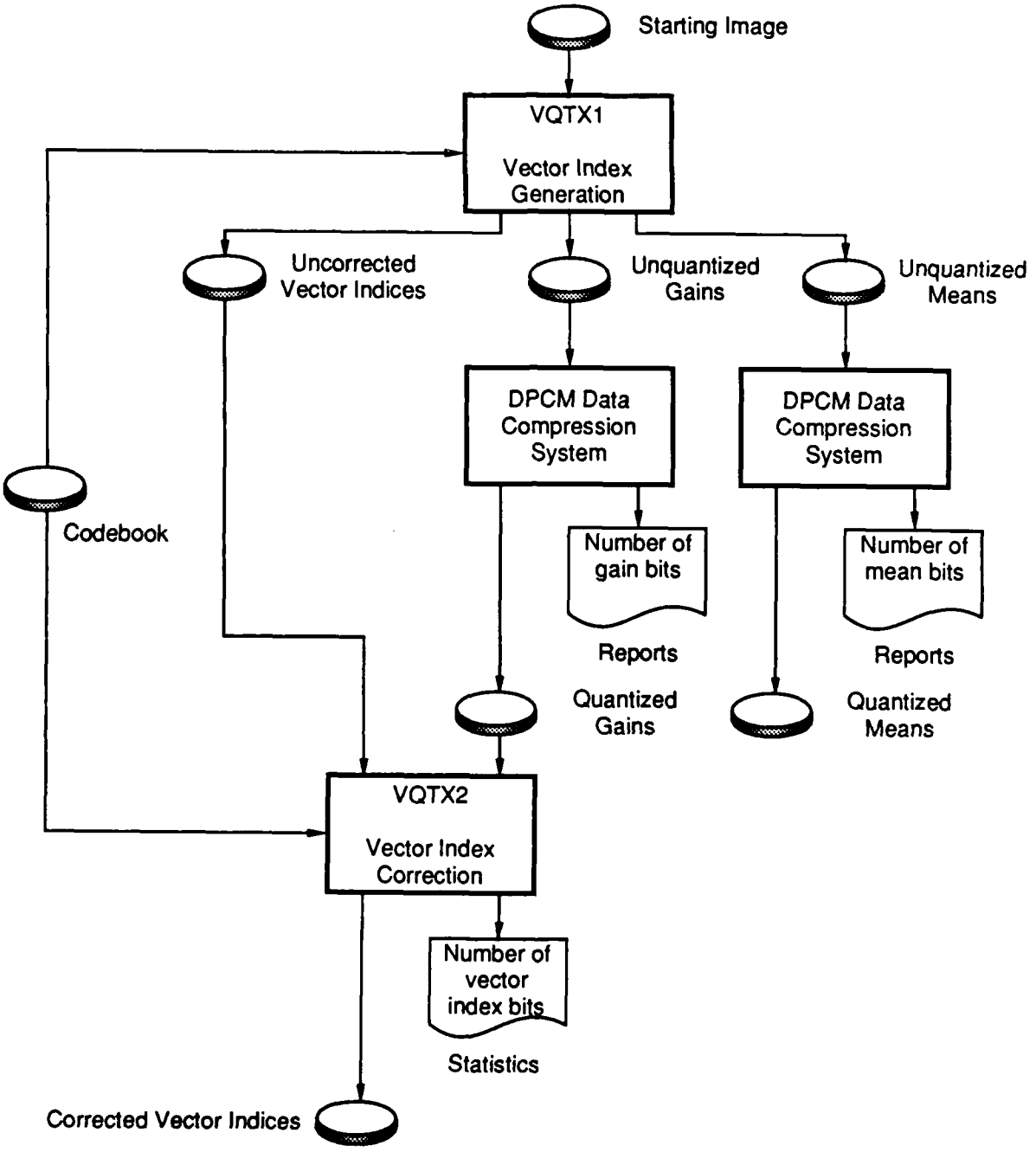


Figure 3.2 Transmitter Simulation

The transmitter transmits three items for each input vector: (1) the vector mean by DPCM, (2) the vector gain by DPCM and (3) the vector's relative codebook vector index by straight binary code, L bits for tree level L, including no bits for tree level 0. The receiver determines the tree level from the gain and a list of gain thresholds in the receiver's codebook. From the tree level the receiver determines the base codebook vector index which, with the received relative index (known to be 0 if the tree level is 0), yields the absolute index.

Program VQTX1 extracts the vector means and writes them to an unquantized mean file as 8-bit "pixels," with one pixel for each vector mean value, i.e., per starting image block. VQTX1 then performs an amplitude-adaptive tree search and writes the gain of each block to an unquantized gain file, also as 8-bit "pixels" VQTX1 then writes the relative codebook vector index of each block to an uncorrected vector index file; VQTX1 also writes the tree level to the index file for vector index correction.

The DPCM system processes the mean and gain "images" as it would any other image and produces quantized mean and gain files, also in image format.

Program VQTX2 performs the codebook vector index correction. This is necessary because the receiver receives a quantized gain value for each vector. The quantized value may be different from the unquantized value, and, in particular, may be on the opposite side of the gain threshold that determined the tree search ending level. VQTX2 determines the tree level expected by the receiver

and, if this level differs from the original level, VQTX2 finds a codebook vector in the new tree level and "transmits" that vector's relative codebook vector index instead of that of the original codebook vector. Without vector index correction, the receiver would not only receive the wrong index, it would expect the wrong number of bits, thus throwing the rest of the transmission out of synchronization.

Figure 3.3 shows the receiver simulator. For each vector, the receiver determines the tree level from the received gain value and the gain threshold table in the receiver's codebook. It then determines the codebook vector base index from the tree level and adds the received relative index to arrive at the absolute index. The receiver retrieves the codebook vector, multiplies it by the gain and adds the mean to each component to produce an output vector. Finally, it writes the output vector as a block to the ending image file.

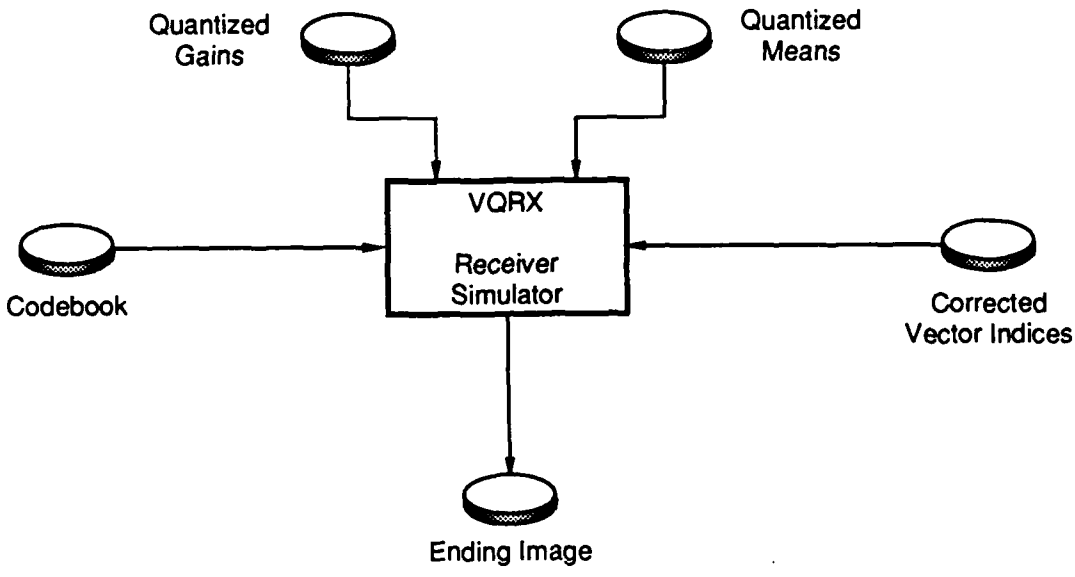


Figure 3.3 Receiver Simulation

4.0 Detailed System Description

The DIS Vector Quantization system implementation is based on the Japan Annex 4 method, but is not necessarily the same as that used by its authors. The Japan Annex 4 paper is very terse, giving only the basic concepts of a codebook containing normalized vectors, transmitting the block mean and gain values by DPCM, and the amplitude-adaptive tree search. The authors touch on codebook generation by mentioning the LBG clustering algorithm, but give no details. The algorithms employed in the DIS simulation software may therefore contain novel features.

4.1. The Codebook

The codebook data structure is central to the entire system. The codebook contains the vector library, the search tree and several small data arrays indexed by tree level.

4.1.1. Codebook Vectors

The codebook vectors are normalized, as they are in the Japan Annex 4 method. In the DIS implementation, each vector component occupies one byte of memory. Because one component of a normalized vector has the range -1 to 1, the usual representation would be by a real (as opposed to integer) number, which normally requires 4 bytes. In the current implementation, the normalized component value is scaled so that the one-byte

integer value contains 8 significant bits of fractional data including sign. While introducing slight rounding errors (the vector is not quite zero-mean and not quite normalized), this approach reduces the codebook memory requirements by a factor of 4. The rounding "noise" was determined to be well below the noise introduced by vector quantization.

In addition to the normalized vector itself, each codebook vector record contains a pointer into the search tree. This pointer is required for vector index correction (described later).

4.1.2. Search Tree and Codebook Partitioning

Figure 4.1 shows the first three levels of a search tree. Each tree node has a pointer to its parent, to its left child (node below and to the left in the figure), and to its right child. (The tree root node has no parent, and the tree "leaves" have no children.)

Each search tree node also contains the relative codebook vector index of the codebook vector belonging to that node. The lowest index is 0, and the highest is 2^L-1, where L is the tree level. Thus, the relative codebook vector index for tree level L is an L-bit binary number. (If a tree node is deleted, as explained later, its index is not used.)

The codebook vectors are partitioned into search tree levels; that is, all the codebook vectors for one search tree level are stored contiguously. Tree level 0 (the tree root) has

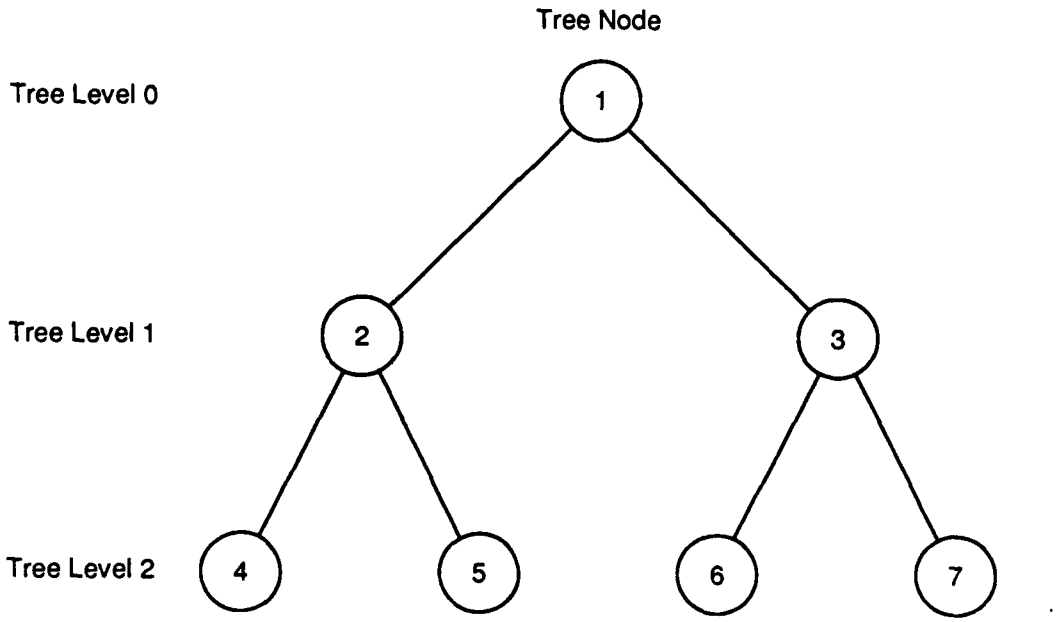


Figure 4.1 Codebook Search Tree

one codebook vector, tree level 1 has two, tree level 2 has four, and so on. The codebook contains an auxiliary array giving the base codebook vector index for each search tree level. The absolute index to a codebook vector for any tree node is computed by adding the relative codebook vector index for the node to the base codebook vector index for the node's tree level, as shown in Figure 4.2.

Each codebook vector has a pointer to the tree node to which it is assigned. Thus, there is a two-way link between each tree node and its codebook vector.

4.1.3. Thresholds

The amplitude-adaptive search and variable-length vector index codes described in the Japan Annex 4 reference imply the existence of gain ("amplitude deviation") thresholds associated with the search tree levels. The reference includes a table of vector index codeword length vs. gain ("amplitude deviation") range. The reference gives no description of how the gain ranges were derived.

In the DIS implementation, the codebook generator automatically derives thresholds that control the tree searches. The thresholds are derived from a single user input: a signal-to-noise ratio goal expressed in decibels.

Figure 4.3 shows the threshold principle. Figure 4.3(a) illustrates the relationship among a starting zero-mean (difference) vector to be quantized, the best codebook vector

Tree Level	Base Codebook Vector Index	Relative Codebook Vector Index	Absolute Codebook Vector Index
0	0	0	0
1	1	0	1
		1	2
2	3	0	3
		1	4
		2	5
		3	6

Figure 4.2 Codebook Vector Indices

for that starting vector and the ending zero-mean vector constructed by the receiver. The page represents the plane determined by the starting and ending vectors and the coordinate system origin. As the figure shows, the gain (dot product of the starting vector and the (normalized) codebook vector) is the magnitude of the ending zero-mean vector.

The angle between the starting vector and the selected codebook vector is a measure of the correlation between the two vectors. The better the correlation (larger the dot product), the smaller the angle. The more densely populated the vector space is by codebook vectors in the vicinity of the starting vector, the smaller the angle will be.

Figure 4.3(b) illustrates the principle of the amplitude-adaptive tree search. For a given distortion, the greater the starting vector magnitude, the greater the required correlation with the codebook vector (the smaller the angle). Therefore, the greater the magnitude, the more densely the codebook vector space must be populated to yield a given distortion. Thus, the codebook is in effect partitioned by tree level. There are twice as many codebook vectors for tree level L+1 as there are for level L. A gain threshold is assigned to each tree level to control where the tree search stops. If the gain is at or below the threshold, the search stops at this level.

Note that Figure 4.3(b) refers to magnitudes, not gains; yet the thresholds are based on gain. This is because the gain, not

the starting vector magnitude, is transmitted. The receiver does not "know" the starting vector magnitude. The gain is transmitted because the distortion is less than if the magnitude were transmitted, as is shown in Figure 4.4. (Gain value error due to DPCM transmission is neglected here for simplicity.) The initial tree search is based on the starting vector magnitude, and a correction is later made to base the final selection on gain.

4.2. Codebook Generator

4.2.1. Training Vectors

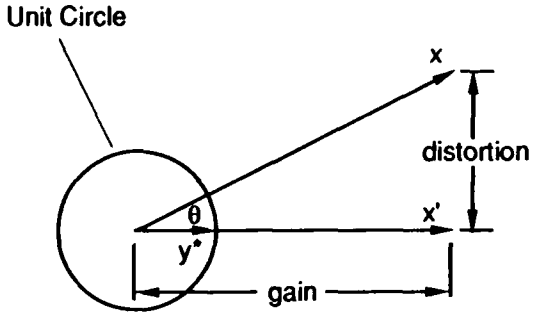
The training vectors are generated by two preprocessing steps prior to codebook generation. The process begins with a training image, which is a composite of selected portions of three of the four standard gray scale images.

The training vector generator builds the training vectors from the image. Each image block is converted to vector form, the mean is subtracted from each component, and the resulting zero-mean vector is normalized. ("Flat" blocks are discarded because zero-magnitude vectors cannot be normalized.) The normalized vector and the magnitude of the unnormalized zero-mean vector are written to a training vector file.

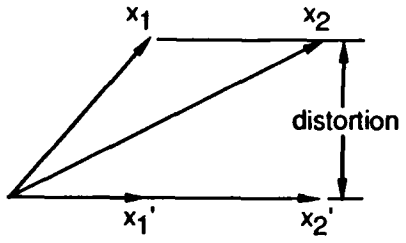
The training vector generator generates one training vector per image block and per each possible offset of the block with respect to the image boundaries. Thus, training image features are represented in all possible positions in a block.

Legend:

x	Starting zero-mean vector
x'	Ending zero-mean vector
y^*	Codebook vector
θ	Angle between starting and ending vector

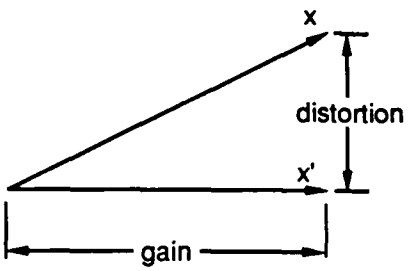


(a) Illustration of Terms

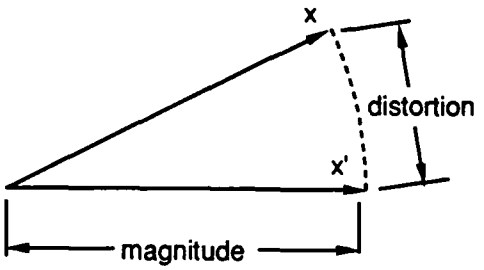


(b) Different Magnitudes, Same Distortion

Figure 4.3 Principles of Magnitude and Gain Thresholds



(a) Transmitting Gain



(b) Transmitting Magnitude

Figure 4.4 Transmitting Gain vs. Magnitude

Next, the training vectors are sorted into descending order of (unnormalized vector) magnitude for reasons to become clear momentarily. The sorted training vector file is the input to the codebook generator.

4.2.2. General Description

The codebook generator starts by building the root tree node. It then "bootstraps" the rest of the codebook. After each tree level is built and optimized, a gain threshold is assigned to that level based on the user-supplied signal-to-noise ratio goal and the mean square error of the training vectors (mean square distance between the training vectors and the codebook vectors to which they belong). Codebook generation terminates when (1) the number of tree levels is such that, at the highest level, the largest possible gain falls at or below the gain threshold, or (2) tree growth has reached a limit based on available memory. When the tree size is limited by available memory, the codebook is suboptimal, and the signal-to-noise ratio is degraded.

4.2.3. Signal-to-Noise Ratio Goal

The codebook generator starts by soliciting a signal-to-noise ratio goal expressed in decibels with reference to the highest gray level (255). The higher the goal, the better the transmitted image quality, but at the expense of compression. Subject to memory limitations, the higher the goal, the larger

5.1.the tree, the greater the number of codebook vectors, and the longer the codebook generator run. The codebook generator converts the dB value to an absolute square error, i.e., the square of the distance between a starting and ending zero-mean vector. This is called the vector mean square error goal.

4.2.4. Tree Growth

Figure 4.5 shows a part of the search tree during codebook generation. Assume that level L-1 has just been optimized and that level L is about to be optimized.

Gain thresholds have already been assigned for tree levels 0 through L-1. Tree level 0 has a low threshold, level 1 has a higher threshold, and level L-1 has the highest so far.

The parent node, P, has a codebook vector that is optimum for tree level L-1. The left child node, C, has a copy of that codebook vector, and the right child node, S, has a "split" vector. The split vector is a copy of that training vector belonging to node P which is nearest, but different from, P's codebook vector. This implementation of vector splitting is a heuristic method of creating a nearby normalized vector with the aim of dividing the training vectors belonging to node P into two roughly equal portions between nodes C and S.

Training vectors are read from the sorted training vector file. For each training vector a tree search is performed exactly as in the transmitter. At each tree level the training vector magnitude (saved in the record along with the

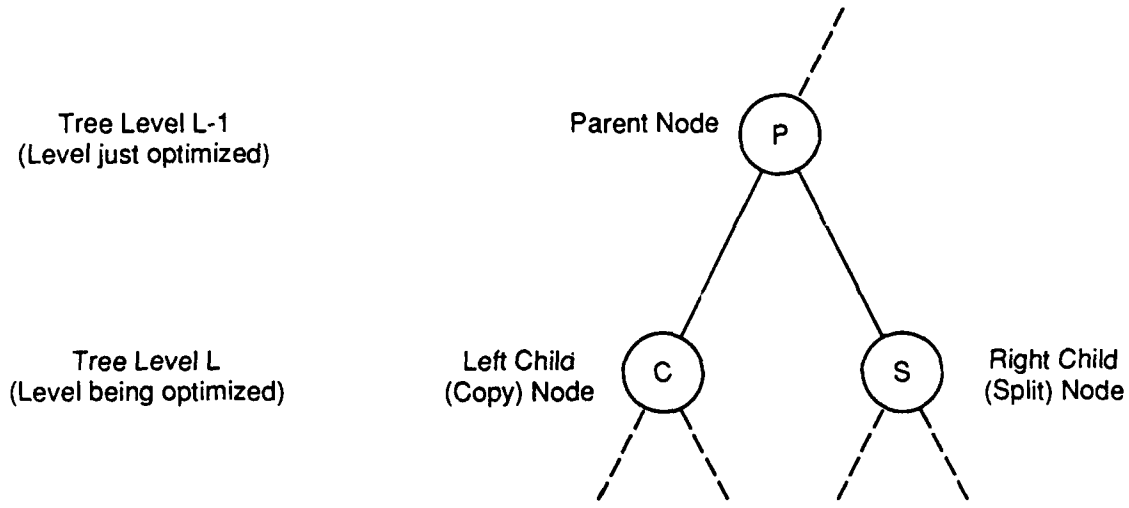


Figure 4.5 Part of Developing Codebook Search Tree

normalized vector) is compared to the gain threshold (actually, the magnitude squared is compared to a magnitude squared threshold).

If the magnitude is above the threshold, the program determines which child of the current node has the best-matching codebook vector (which codebook vector has the larger dot product with the training vector). The search progresses to the better node. Ties are broken arbitrarily in favor of the left child.

If the training vector magnitude is at or below the gain threshold for the current level, or if that level is level L, the search halts. When a training vector tree search ends at a node in level L, its components are added to a centroid sum vector associated with that node. The square error with respect to the node's codebook vector is computed and added to an accumulating new square error sum. If the error is not zero, but smaller than any previous error for that node, then a copy of the training vector replaces the codebook vector for the right child of the node, i.e. in tree level L+1, if memory permits a higher level. When level L is optimized, split vectors will exist at tree level L+1.

If the search halts at a lower level than level L, no more training vectors need be read. This is because the vectors are sorted in descending order of vector magnitude. If any training vector fails to reach level L, all those following it in the file would also fail. Without the sort, the program would have to test <u>all</u> the training vectors and ignore those that do not reach

level L.

After all training vectors reaching level L have been processed, an update is performed. For each node P in level L-1 the total error for the two child nodes is compared with the total for the previous training vector pass. (The first "previous" error is set to "infinity.") If the error has decreased significantly, the two child nodes are still suboptimal; otherwise they are flagged "optimal," and training vectors arriving at these nodes are ignored in future passes. If the two nodes are still suboptimal, then the centroid sum vector for each of the two nodes is normalized and copied into the codebook, replacing the previous codebook vectors. This is equivalent to updating the codebook with the centroid, as in the LBG clustering algorithm, with the constraint that all codebook vectors are normalized. The accumulated new square error for the two nodes becomes the old square error, and the new square error is reset to zero.

If some tree nodes in level L have not yet been optimized, then the training vectors are processed again, and another update is performed. This cycle is repeated until all level L nodes are optimized or until the maximum allowed number of iterations (program parameter) have been performed. In the latter case, all remaining unoptimized nodes are declared optimal, with consequent degradation in codebook performance.

When level L has been optimized, a tree level wrapup is performed. The mean square error for all nodes in the level is

computed. Because the training vectors, as well as the codebook vectors, are normalized, and because negative gains are set to 0, the mean square error has a range of 0 to 1.0 inclusively. This mean square error is therefore called the normalized mean square error. A magnitude squared threshold is computed by dividing the vector mean square error goal by the normalized mean square error for the tree level. The gain threshold is the square root of this number. Both values are stored in small codebook arrays.

4.2.5. Zero-Error Nodes

A zero-error tree node is defined as a tree node either to which no training vectors belong, or all training vectors belonging to it are perfectly correlated with the node's codebook vector. In either case, the total square error for the node is zero. Zero-error nodes can occur at high tree levels, because relatively few training vectors reach these higher levels.

When a zero-error node is detected, its right child, if one exists, is deleted. The deleted node is where the zero-error node's split vector would have been maintained. Since, by definition, the zero-error node has no error, there is no split vector. If tree growth progresses past the lowest level containing a zero-error node, the zero-error node's codebook vector is copied to its left child. This node, too, becomes a zero-error node because the training vectors reaching it comprise a subset of those reaching its parent node. Since the training vectors reaching the parent zero-error node produced no error,

there can be none at the child node.

4.3. Transmitter Simulator

4.3.1. VQTX1 - Vector Index Generation

For each starting image vector (block) VQTX1 computes the vector mean and writes it to a "mean image" for processing by the DIS DPCM data compression system, and subtracts the mean from each vector component to yield a starting zero-mean vector.

VQTX1 next performs a preliminary binary tree search. At each stage, the choice of the left or right child tree node is based on which codebook vector has the larger dot product with the zero-mean starting vector. The search terminates at the lowest tree level for which the magnitude of the zero-mean input vector is at or below the gain threshold for that level. (Actually, the square of the magnitude is compared with a stored table of magnitude-squared thresholds.) As the search progresses, the gains and relative codebook vector indices for all lower levels are saved in a "stack" for reasons to be explained next.

VQTX1 tests whether the gain for the preliminary search ending level lies above the gain threshold of the next lower tree level. If so, the final search result is the same as the preliminary. If not, the ending level is backed up to the next lower level, the gain and relative codebook vector index are replaced by the values saved in the "stack" at the lower level, and the test is repeated. The final search ends when the

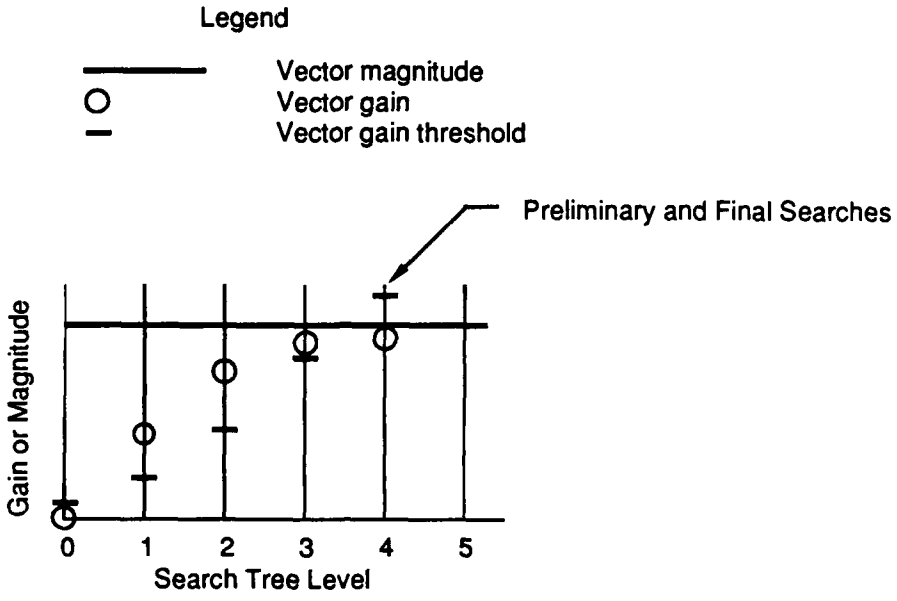
gain at the ending level is above the gain threshold for the next lower level. If the stack is exhausted without this condition's being met, the ending level is level 0. This final search is needed because the gain, not the magnitude, is transmitted to minimize distortion, as shown in Figure 4.4

Figure 4.6 illustrates why the preliminary search is based on the starting (zero-mean) vector magnitude instead of gain, and why the final search sometimes ends at a different tree level from that of the preliminary search.

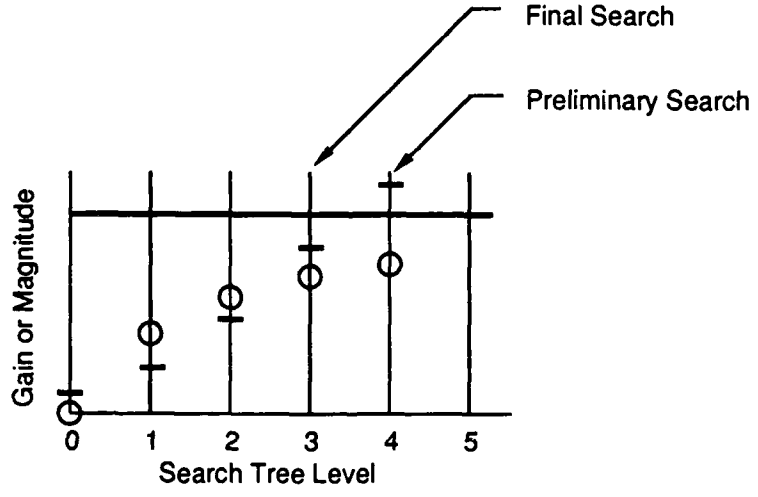
Were the preliminary search based on gain, the search might stop at a low tree level, e.g. 0, where the correlation of the input vector with a codebook vector is so poor that the gain is zero, even for an input vector of substantial magnitude. (Negative gains are set to zero.) The result would be to replace a starting block containing considerable detail with a "flat" block, all of whose pixel gray levels are that of the mean.

The final, backward, search is needed if, at the lowest tree level where the magnitude is at or below the gain threshold, the gain is at or below the gain threshold at the next lower level. Since the gain, not the magnitude, is transmitted, the receiver would "think" the search ended at a lower tree level, and would expect the wrong number of bits for the relative codebook vector index. Figures 4.6(a) and 4.6(b) illustrate the preliminary and final searches ending at the same and different tree levels respectively.

The tree search complete, VQTX1 writes the gain to a "gain



(a) Final Search Same As Preliminary



(b) Final Search Different From Preliminary

Figure 4.6 Illustration of Transmitter Tree Search

image" similar to the mean image for "transmission" by DPCM. It also writes to an uncorrected vector index file a vector index record containing the tree level and the relative codebook vector index. The level is not "transmitted," but is used by program VQTX2 for vector index correction, and by the receiver simulator as a program check to verify that the receiver expects the same number of relative codebook vector index bits as were "transmitted."

4.3.2. Mean and Gain Image Processing

Transmission of the vector means and gains is simulated by separate executions of the existing DIS DPCM compression system. The DPCM simulator, employing prediction, a 3-level quantizer and entropy coding, is a stand-alone program that operates on an entire image. Program VQTX1 generates separate "unquantized" gain and mean "images" each containing one pixel per starting image block. Each "image" is processed separately through the DPCM simulator, which produces a "quantized image" and reports the number of bits required to "transmit" the "image." The resulting "quantized" mean and gain "images" are approximations of the unquantized "images."

4.3.3. VQTX2 - Vector Index Correction

Program VQTX2 corrects and "transmits" the relative codebook vector indices. Correction consists of determining from the received gain value (after DPCM) the tree level at which the

receiver "thinks" the tree search ended, which is also the number of relative codebook vector index bits the receiver expects. This tree level is the lowest for which the received gain is at or below the level's gain threshold. If this perceived level is the same as the level where the final VQTX1 search ended, then VQTX2 writes the level number and uncorrected relative codebook vector index to a corrected vector index file. Otherwise, VQTX2 reenters the search tree via the pointer contained in the codebook vector VQTX1 selected, and searches upward or downward in the search tree as needed to find a codebook vector at the correct tree level. VQTX2 writes the correct level and relative codebook vector index to the corrected vector index file. The relative codebook vector index is "transmitted" as a straight binary number without any further encoding.

Vector index correction is required mainly because the DPCM "transmission" of a gain value might put the received gain value on the wrong side of a gain threshold, making the receiver expect the wrong number of bits. Another reason for correction is the fact that, while the correlation of an input vector with a codebook vector improves on the average with increasing tree level, this is not necessarily the case for an individual vector. Occasionally, the preliminary tree search is forced to a higher tree level because the vector magnitude is above the current level's gain threshold, even when the gain is less at the higher level than at the lower. When this happens, the final search may select a lower tree level, but in this particular case, the

ending gain is above the gain threshold. VQTX2 detects the fact that the receiver "thinks" the search ended at a higher level and makes the correction.

4.3.4. Transmission Statistics

Program VQTX2, besides correcting and "transmitting" the relative codebook vector indices, compiles and reports statistics for system performance evaluation. The statistics report includes: (1) the number of pixels per block; (2) the total number of blocks "transmitted;" (3) the total number of codebook vector indices "transmitted" (No index is "transmitted" if the corrected tree level is zero.); (4) the total number of codebook vector index bits "transmitted;" (5) the average number of codebook vector index bits per pixel by the present method, by minimum-entropy coding of the relative codebook vector indices and by minimum-entropy coding of the absolute codebook vector indices; (6) a table showing, for each search tree level, the number of tree searches ending at that level after correction, the codeword length for that level (equals the level number) and the theoretical minimum-entropy codeword length for that level based on the statistics of the current transmission; and finally, (7) the number of vector index corrections to higher and to lower tree levels.

4.4. Receiver Simulator

The receiver simulator, VQRX, is trivial. It builds the output image from the DPCM-quantized mean and gain images and from the corrected codebook vector index file. It also performs a system error check.

4.4.1. Image Reconstruction

Program VQRX "receives" each block one at a time by reading a mean from the quantized mean "image," a gain from the quantized gain "image" and a relative codebook vector index from the corrected vector index file.

VQRX searches the gain threshold table in the codebook to determine the lowest tree level for which the received gain is less than or equal to the threshold. The level number is the number of bits, including none for level 0, the receiver expects to receive for the relative codebook vector index. VQRX then looks up the codebook vector by adding the relative codebook vector index (known to be 0 if the level is 0) to the base codebook vector index for the level. VQRX multiplies each codebook vector component by the received gain and adds the received mean. Finally, it writes the resulting output vector, reorganized as an image block, to the output image.

4.4.2. System Error Check

Instead of actually assembling and disassembling a bit stream, the simulation system simply writes and reads the data

for each image block to and from files. DPCM transmission of the means and gains is simulated by a separate stand-alone program as explained above.

The system does, however, have a safeguard against the possibility of a design or programming error's being hidden by the lack of a bit stream. Each record of the corrected codebook vector index file contains two items: (1) the relative codebook vector index, which would be transmitted in an actual system, and the tree level number to which that index pertains. This level number would not be transmitted; it is provided for program checking purposes only. The receiver simulator determines the level number, L, from the gain and the gain threshold table contained in the codebook. It then checks that L is equal to the level number contained in the record, and that the relative codebook vector index can be expressed in L bits. The possibility of transmission errors is ignored because Group 4 facsimile transmissions are guaranteed to be error-free.

5.0 Simulation

5.1. Image Selection Criteria

5.1.1. Test Images

The selection of the test images employed in the simulations was based on several factors, including image quality, availability and feature content. The test images were the four standard gray scale images developed by DIS for the NCS in a previous study and since adopted by the CCITT. The standard gray scale image selections were based on a set of characteristics designed to thoroughly test various gray scale transmission techniques.

Beyond the advantages these images provide in terms of image quality and availability, each image was selected because it contained several distinctive features that would aid in the subjective evaluation of the output images. The IEEE image is representative of an identification card, combining both photographic and textual information, and includes a high contrast wedge that aids in the evaluation of an algorithm's effect on resolution. The house and sky image contains large areas of gradually changing gray scale, several areas of varying texture, and various horizontal, vertical and diagonal lines. The house with trees image is similar, but also contains high detail regions. The aerial photograph is a low contrast image of high detail and relatively low resolution.

5.1.2. Training Image

The training image is a composite of three of the four test images. The IEEE image was deliberately omitted so any tendency for a codebook to "memorize" the training image would be revealed by better performance with the three images from which the training image was composed than with the fourth. The training image contains three broad, horizontal stripes, each representing mostly "busy" parts of one of the three test images. "Busy" (high detail, high contrast) parts of the images were chosen to produce high-magnitude vectors that, during codebook generation, would propagate to the highest search tree level.

Choosing a broad stripe from each of the three contributing test images allows further evaluation of the "memorization" problem. If memorization were significant, one would observe in an encoded image a stripe of improved quality where the test image matches the training image stripe. The actual simulations showed no evidence of memorization.

5.2. Simulation Parameters

The following simulation parameters were maintained constant throughout all the tests except where otherwise noted.

Block Size

The block size is 16. Each block is four pixels wide by four high. This is a program parameter, and can easily be changed.

Mean and Gain Transmission

The means and gains were "transmitted" by PCM, not DPCM as originally intended, at a total cost for both of 16 bits per block, or one bit per pixel. Initial tests showed that the mean and gain "image" statistics were sufficiently different from those of ordinary images that the existing DIS predictor and 3-bit quantizer introduced excessive distortion, far more than did quantization of the residue vectors. Tailoring the DPCM parameters to mean and gain "images" was deferred in favor of evaluating vector quantization itself. Thus, the "quantized" mean and gain "images" were actually the unquantized images generated by program VQTX1.

Maximum Search Tree Level

Encoding of the house and sky image was simulated with codebooks generated with a maximum search tree level of 11 (4095 codebook vectors) and 13 (16383 codebook vectors) to evaluate the effect of increased codebook capacity on image quality and data compression with other parameters held constant. If the signal-to-noise ratio goal, specified when a codebook is generated, is sufficiently high that tree growth terminates because of memory limits set by the maximum tree level, then image quality is degraded in areas of high contrast.

Number of Training Vectors

The training vector file contains all the training vectors derived from the training image (65000) sorted in descending order of vector magnitude, i.e., the highest contrast blocks come first. The codebook generator contains a parameter that specifies how many, if not all, of these training vectors are read during any one iteration. Because codebook generation runs are long (hours of execution time), the codebooks used in these simulations were derived from much fewer than 65000 training vectors. Most of the simulations were performed with 12000, 6000, and 3000. It is important to note that selecting the first N training vectors selects the N vectors having the largest magnitudes, not those from a specific part of the training image.

Signal-To-Noise Ratio Goal

The signal-to-noise ratio goal, entered during codebook generation, is the variable to which the tradeoff between image quality and data compression is most sensitive. Simulations were performed with values of 50, 40 and 30 dB.

Image Resolution

The resolution of the training image and all test images is 200 pixels per inch.

Image Size

All test images are 512 pixels wide by 512 high. These dimensions were selected to facilitate the display, photographing, and printing of the output images.

5.3. Evaluation Criteria

Results were evaluated for data compression, RMS error and subjective image quality.

Data compression is expressed in bits per pixel. In all cases the values were 1.x, where the 1 is the cost, for a 16-pixel block, of transmitting the block mean and gain by PCM, and the x is the cost (less than one bit per pixel) of transmitting the relative codebook vector index.

The RMS error is the root-mean-square-error of the encoded image pixels when compared to the corresponding pixels of the original test image.

The encoded images were evaluated subjectively and given subjective image quality ratings (SIQR) as defined below:

Rating	Definition
0	Image is not recognizable.
1	Almost no detail is evident; only general outlines of objects remain.
2	Loss of edge detail almost total; objects in image unrecognizable.
3	Image is slightly recognizable; edge boundaries severely distorted.
4	Image is partially recognizable; complete loss of detail in several image regions is evident.
5	Image is recognizable, but shows severe blocking artifact throughout and poor detail rendition.

6	Blocking is severe in regions of high detail, and detail rendition is marginal.
7	Blocking is moderate in high-detail areas, and detail rendition is fair.
8	Blocking is slightly evident, and detail rendition is good.
9	No blocking is evident, and detail rendition is very good.
10	Encoded image is indistinguishable from original image.

The subjective image quality evaluation was performed using both photographic and video display representatives of the output images. The photographs were produced by using a high quality 4 x 5 camera to photograph each image displayed on a high resolution gray scale monitor.

5.4. Results

The house and sky image simulations were performed using the various combinations of maximum tree level, number of training vectors and codebook signal-to-noise ratio goals. The purpose of these simulations was to explore, with one test image, the system performance over a wide range of parameters to establish parameters for the remaining simulations. The results are shown in Table 5.1.

These preliminary simulations showed that there was insignificant change in data compression and image quality when the maximum tree level was increased from 11 to 13. The remaining simulations were therefore performed with codebooks

Number of Training Vectors

S/N Ratio Goal (dB) \	1500	3000	6000	12000
50	B/P: 1.24 RMS: 2.94 SIQR: 5.0	B/P: 1.25 RMS: 2.61 SIQR: 9.0	B/P: 1.27 RMS: 2.39 SIQR: 9.3	B/P: 1.29 RMS: 2.19 SIQR: 9.5
40				
30			B/P: 1.03 RMS: 4.31 SIQR: 6.5	B/P: 1.03 RMS: 4.33 SIQR: 6.5

Image: House and Sky

Legend:

B/P Bits per Pixel

RMS RMS Error

SIQR Subjective Image Quality Rating

(a) Maximum Codebook Tree Level = 13

Number of Training Vectors

S/N Ratio Goal (dB) \	1500	3000	6000	12000
50		B/P: 1.23 RMS: 2.63 SIQR: 8.5		B/P: 1.26 RMS: 2.29 SIQR: 9.4
40				
30				B/P: 1.03 RMS: 4.33 SIQR: 6.5

(b) Maximum Codebook Tree Level = 11

Table 5.1 Varying Number of Training Vectors and Maximum Codebook Tree Level

generated with the maximum tree level set at 11, i.e., a maximum of 4095 codebook vectors.

Encoding and decoding of all four test images were simulated with codebooks generated with all nine combinations of 12000, 6000 and 3000 training vectors and signal-to-noise ratio goals of 50, 40 and 30 dB, for a total of 36 simulations.

Table 5.2 shows the compressed bit rates, RMS errors and subjective image quality ratings (SIQR) for the IEEE face image for all nine codebooks. Tables 5.3, 5.4, and 5.5 show similar data for the house and sky, house with trees, and aerial photograph images. These tables show how a given image is affected by the various codebook generation parameters.

Table 5.6 shows the same data for all four images with the number of training vectors held constant at 12000. The figure numbers under the image titles are those of photographs of the encoded and decoded images.

Tables 5.7 and 5.8 are similar, with the number of training vectors constant at 6000 and 3000 respectively.

Table 5.9 (two pages) is an index to the photographs of all original and processed images. The photographs are displayed in Figures 5.1 - 5.40.

Each of the four images is displayed in 10 photographs, the first showing the original image (digitized to the 8-bit gray scale), and the other nine showing the results of encoding and decoding the same image using the nine combinations of number of training vectors and signal-to-noise ratio goals.

Legend:

B/P	Bits per Pixel
RMS	RMS Error
SIQR	Subjective Image Quality Rating

Number of Training Vectors

S/N Ratio Goal (dB)

	3000	6000	12000
50	B/P: 1.29 RMS: 2.31 SIQR: 8.5	B/P: 1.31 RMS: 2.13 SIQR: 9.0	B/P: 1.33 RMS: 2.06 SIQR: 9.5
40	B/P: 1.11 RMS: 3.18 SIQR: 6.8	B/P: 1.14 RMS: 2.61 SIQR: 7.8	B/P: 1.16 RMS: 2.52 SIQR: 8.0
30	B/P: 1.02 RMS: 4.57 SIQR: 5.2	B/P: 1.03 RMS: 4.24 SIQR: 5.7	B/P: 1.04 RMS: 4.23 SIQR: 6.2

Table 5.2 Varying Codebook Parameters: IEEE Face

Legend:

B/P	Bits per Pixel
RMS	RMS Error
SIQR	Subjective Image Quality Rating

Number of Training Vectors

S/N Ratio Goal (dB)	3000	6000	12000
50	B/P: 1.23 RMS: 2.63 SIQR: 8.5	B/P: 1.24 RMS: 2.44 SIQR: 8.8	B/P: 1.26 RMS: 2.29 SIQR: 9.4
40	B/P: 1.10 RMS: 3.29 SIQR: 7.0	B/P: 1.12 RMS: 2.84 SIQR: 7.5	B/P: 1.14 RMS: 2.67 SIQR: 8.0
30	B/P: 1.02 RMS: 4.56 SIQR: 5.0	B/P: 1.03 RMS: 4.31 SIQR: 5.5	B/P: 1.03 RMS: 4.33 SIQR: 6.5

Table 5.3 Varying Codebook Parameters: House and Sky

Legend:

B/P	Bits per Pixel
RMS	RMS Error
SIQR	Subjective Image Quality Rating

Number of Training Vectors

S/N Ratio Goal (dB)

	3000	6000	12000
50	B/P: 1.56 RMS: 7.59 SIQR: 8.0	B/P: 1.58 RMS: 7.07 SIQR: 8.4	B/P: 1.59 RMS: 6.70 SIQR: 9.0
40	B/P: 1.34 RMS: 8.83 SIQR: 6.0	B/P: 1.41 RMS: 7.36 SIQR: 6.4	B/P: 1.44 RMS: 6.93 SIQR: 7.3
30	B/P: 1.09 RMS: 11.2 SIQR: 5.0	B/P: 1.13 RMS: 10.2 SIQR: 5.2	B/P: 1.15 RMS: 9.75 SIQR: 5.5

Table 5.4 Varying Codebook Parameters: House with Trees

Legend:

B/P	Bits per Pixel
RMS	RMS Error
SIQR	Subjective Image Quality Rating

Number of Training Vectors

S/N Ratio Goal (dB)	3000	6000	12000
50	B/P: 1.52 RMS: 4.00 SIQR: 7.8	B/P: 1.54 RMS: 3.66 SIQR: 8.3	B/P: 1.56 RMS: 3.43 SIQR: 9.0
40	B/P: 1.30 RMS: 5.00 SIQR: 6.5	B/P: 1.35 RMS: 4.05 SIQR: 7.0	B/P: 1.38 RMS: 3.79 SIQR: 7.5
30	B/P: 1.08 RMS: 7.26 SIQR: 5.0	B/P: 1.11 RMS: 6.63 SIQR: 5.5	B/P: 1.13 RMS: 6.39 SIQR: 6.2

Table 5.5 Varying Codebook Parameters: Aerial Photograph

Constant Parameters

Image Resolution:	200 pixels per inch
Image Size:	512 X 512 pixels
Block Size:	4 X 4 pixels
Maximum Tree Level:	11

Image	Codebook S/N Ratio Goal	Compressed Bits per Pixel	RMS Error	Subjective Image Quality Rating
IEEE Face (Figs. 5.2 - 5.4)	50	1.33	2.06	9.5
	40	1.16	2.52	8.0
	30	1.04	4.23	6.2
House and Sky (Figs. 5.12 - 5.14)	50	1.26	2.29	9.4
	40	1.14	2.67	8.0
	30	1.03	4.33	6.5
House with Trees (Figs. 5.22- 5.24)	50	1.59	6.70	9.0
	40	1.44	6.93	7.3
	30	1.15	9.75	5.5
Aerial Photograph (Figs. 5.32 - 5.34)	50	1.56	3.43	9.0
	40	1.38	3.79	7.5
	30	1.13	6.39	6.2

Table 5.6 Performance and Compression with 12000 Training Vectors

Constant Parameters

Image Resolution:	200 pixels per inch
Image Size:	512 X 512 pixels
Block Size:	4 X 4 pixels
Maximum Tree Level:	11

Image	Codebook S/N Ratio Goal	Compressed Bits per Pixel	RMS Error	Subjective Image Quality Rating
IEEE Face (Figs. 5.5 - 5.7)	50	1.31	2.13	9.0
	40	1.14	2.61	7.8
	30	1.03	4.24	5.7
House and Sky (Figs. 5.15 - 5.17)	50	1.24	2.44	8.8
	40	1.12	2.84	7.5
	30	1.03	4.31	5.5
House with Trees (Figs. 5.25 - 5.27)	50	1.58	7.07	8.4
	40	1.41	7.36	6.4
	30	1.13	10.2	5.2
Aerial Photograph (Figs. 5.35 - 5.37)	50	1.54	3.66	8.3
	40	1.35	4.05	7.0
	30	1.11	6.63	5.5

Table 5.7 Performance and Compression with 6000 Training Vectors

Constant Parameters

Image Resolution:	200 pixels per inch
Image Size:	512 X 512 pixels
Block Size:	4 X 4 pixels
Maximum Tree Level:	11

Image	Codebook S/N Ratio Goal	Compressed Bits per Pixel	RMS Error	Subjective Image Quality Rating
IEEE Face (Figs. 5.8 - 5.10)	50	1.29	2.31	8.5
	40	1.11	3.18	6.8
	30	1.02	4.57	5.2
House and Sky (Figs. 5.18 - 5.20)	50	1.23	2.63	8.5
	40	1.10	3.29	7.0
	30	1.02	4.58	5.0
House with Trees (Figs. 5.28 - 5.30)	50	1.56	7.59	8.0
	40	1.34	8.83	6.0
	30	1.09	11.2	5.0
Aerial Photograph (Figs. 5.38 - 5.40)	50	1.52	4.00	7.8
	40	1.30	5.00	6.5
	30	1.08	7.26	5.0

Table 5.8 Performance and Compression with 3000 Training Vectors

Image	Number of Training Vectors	Codebook S/N Ratio Goal	Image Figure Number
IEEE Face	(Original)	(Original)	5.1
	12000	50	5.2
		40	5.3
		30	5.4
	6000	50	5.5
		40	5.6
		30	5.7
	3000	50	5.8
		40	5.9
		30	5.10
House and Sky	(Original)	(Original)	5.11
	12000	50	5.12
		40	5.13
		30	5.14
	6000	50	5.15
		40	5.16
		30	5.17
	3000	50	5.18
		40	5.19
		30	5.20

(Continued)

Table 5.9 Image Index

Image	Number of Training Vectors	Codebook S/N Ratio Goal	Image Figure Number
House with Trees	(Original)	(Original)	5.21
	12000	50	5.22
		40	5.23
		30	5.24
	6000	50	5.25
		40	5.26
		30	5.27
	3000	50	5.28
		40	5.29
		30	5.30
Aerial Photograph	(Original)	(Original)	5.31
	12000	50	5.32
		40	5.33
		30	5.34
	6000	50	5.35
		40	5.36
		30	5.37
	3000	50	5.38
		40	5.39
		30	5.40

Table 5.9 Image Index (Concluded)

Figure 5.1 - IEEE Face Original Image

Figure 5.2 - IEEE Face Output Image
NTV=12000, SNR=50

Figure 5.3 - IEEE Face Output Image
NTV=12000, SNR=40

Figure 5.4 - IEEE Face Output Image
NTV=12000, SNR=30

Figure 5.5 - IEEE Face Output Image
NTV=6000, SNR=50

Figure 5.6 - IEEE Face Output Image
NTV=6000, SNR=40

Figure 5.7 - IEEE Face Output Image
NTV=6000, SNR=30

Figure 5.8 - IEEE Face Output Image
NTV=3000, SNR=50

Figure 5.9 - IEEE Face Output Image
NTV=3000, SNR=40

Figure 5.10 - IEEE Face Output Image
NTV=3000, SNR=30

Figure 5.11 - House and Sky Original Image

Figure 5.12 - House and Sky Output Image
NTV=12000, SNR=50

Figure 5.13 - House and Sky Output Image
NTV=12000, SNR=40

Figure 5.14 - House and Sky Output Image
NTV=12000, SNR=30

Figure 5.15 - House and Sky Output Image
NTV=6000, SNR=50

Figure 5.16 - House and Sky Output Image
NTV=6000, SNR=40

Figure 5.17 - House and Sky Output Image
NTV=6000, SNR=30

Figure 5.18 - House and Sky Output Image
NTV=3000, SNR=50

Figure 5.19 - House and Sky Output Image
NTV=3000, SNR=40

Figure 5.20 - House and Sky Output Image
NTV=3000, SNR=30

Figure 5.21 - House with Trees Original Image

Figure 5.22 - House with Trees Output Image
NTV=12000, SNR=50

Figure 5.23 - House with Trees Output Image
NTV=12000, SNR=40

Figure 5.24 - House with Trees Output Image
NTV=12000, SNR=30

Figure 5.25 - House with Trees Output Image
NTV=6000, SNR=50

Figure 5.26 - House with Trees Output Image
NTV=6000, SNR=40

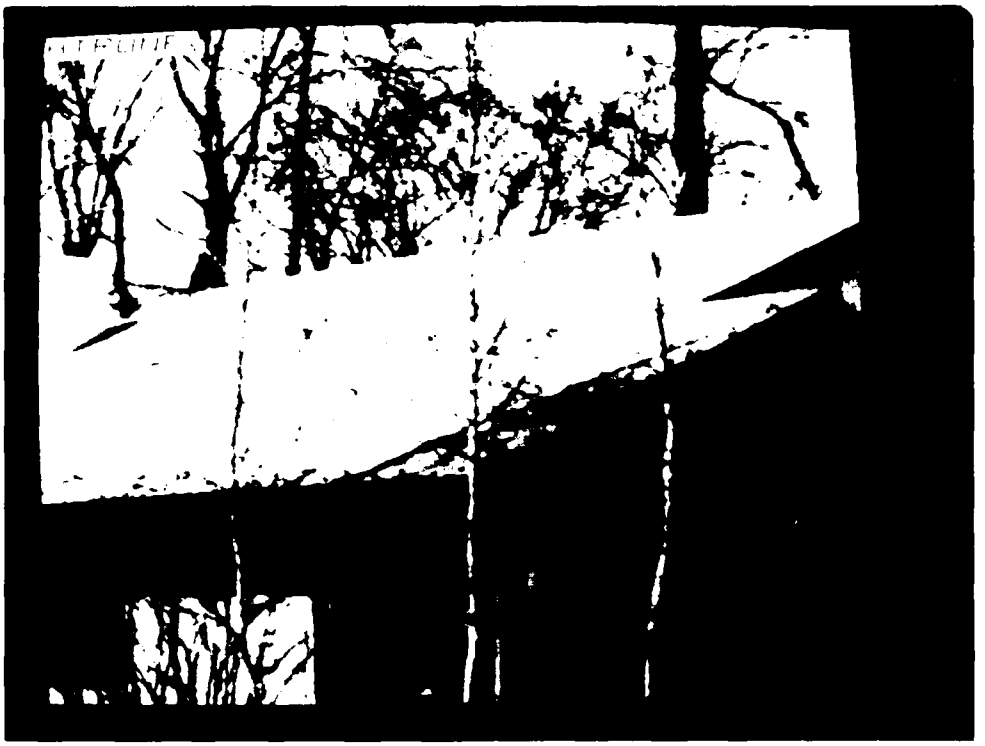


Figure 5.27 - House with Trees Output Image
NTV=6000, SNR=30

Figure 5.28 - House with Trees Output Image
NTV=3000, SNR=50

Figure 5.29 - House with Trees Output Image
NTV=3000, SNR=40

Figure 5.30 - House with Trees Output Image
NTV=3000, SNR=30

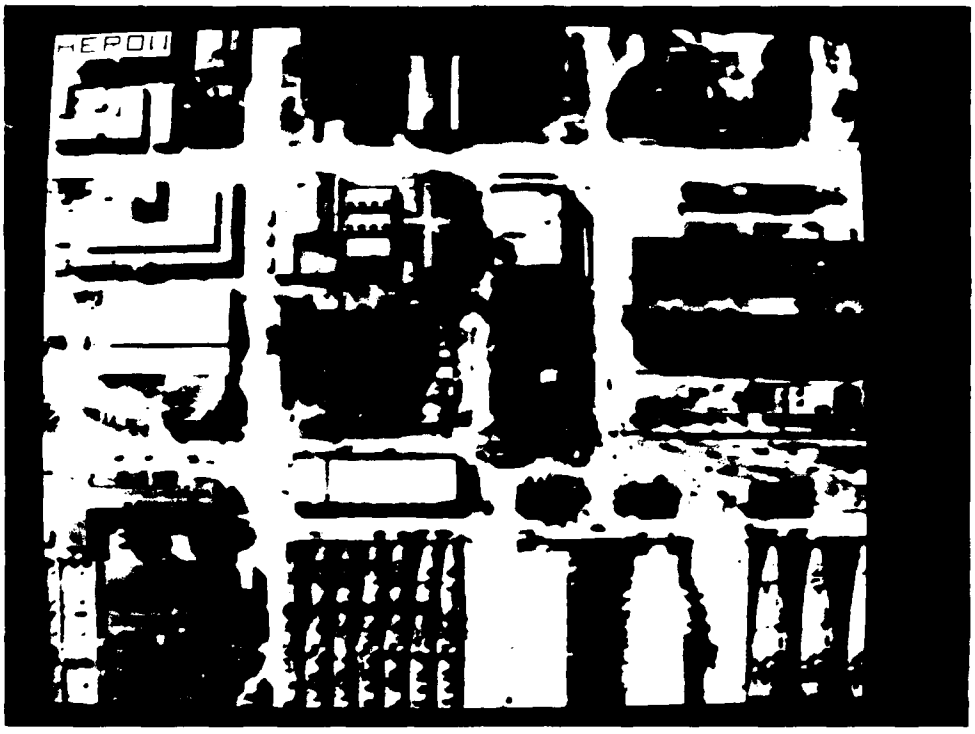


Figure 5.31 - Aerial Photo Original Image

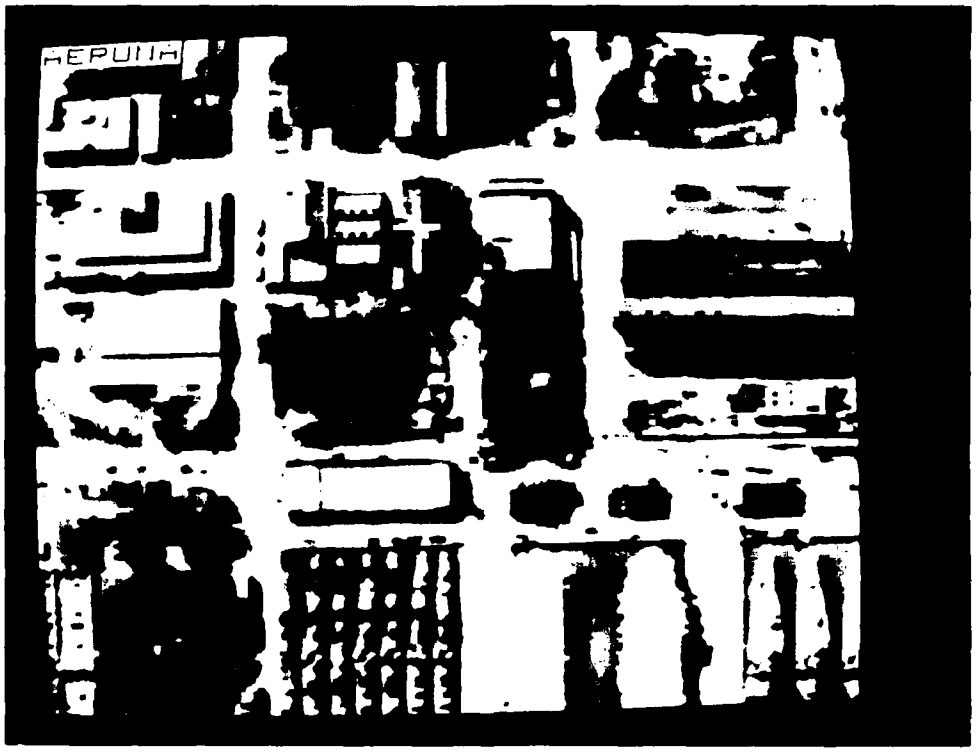


Figure 5.32 - Aerial Photo Output Image
NTV=12000, SNR=50

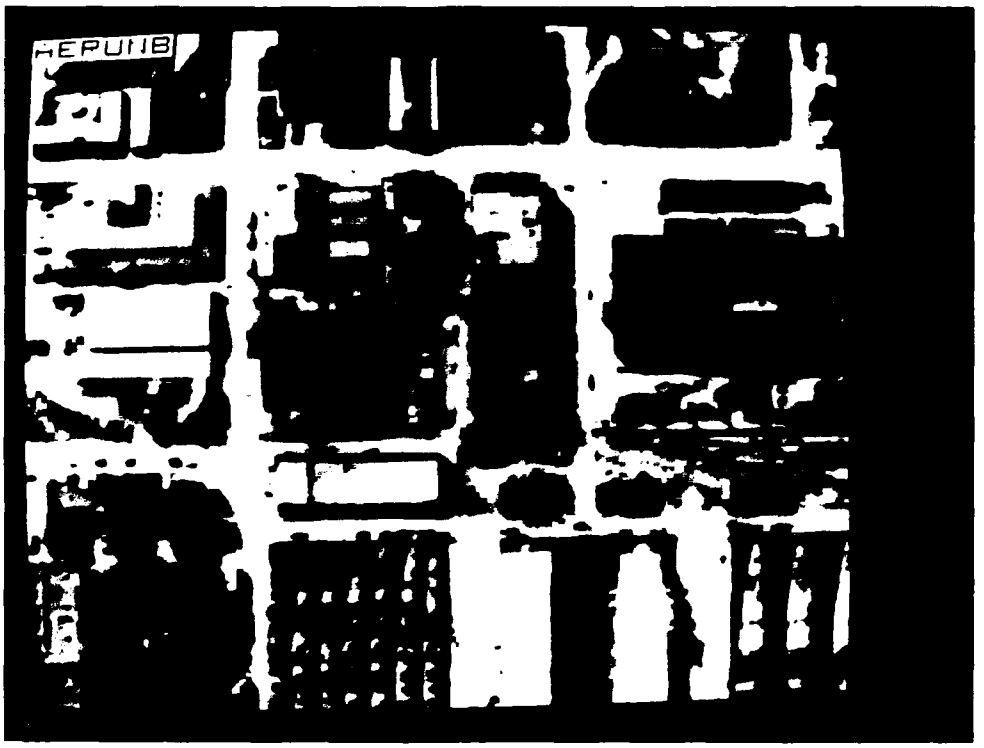


Figure 5.33 - Aerial Photo Output Image
NTV=12000, SNR=40

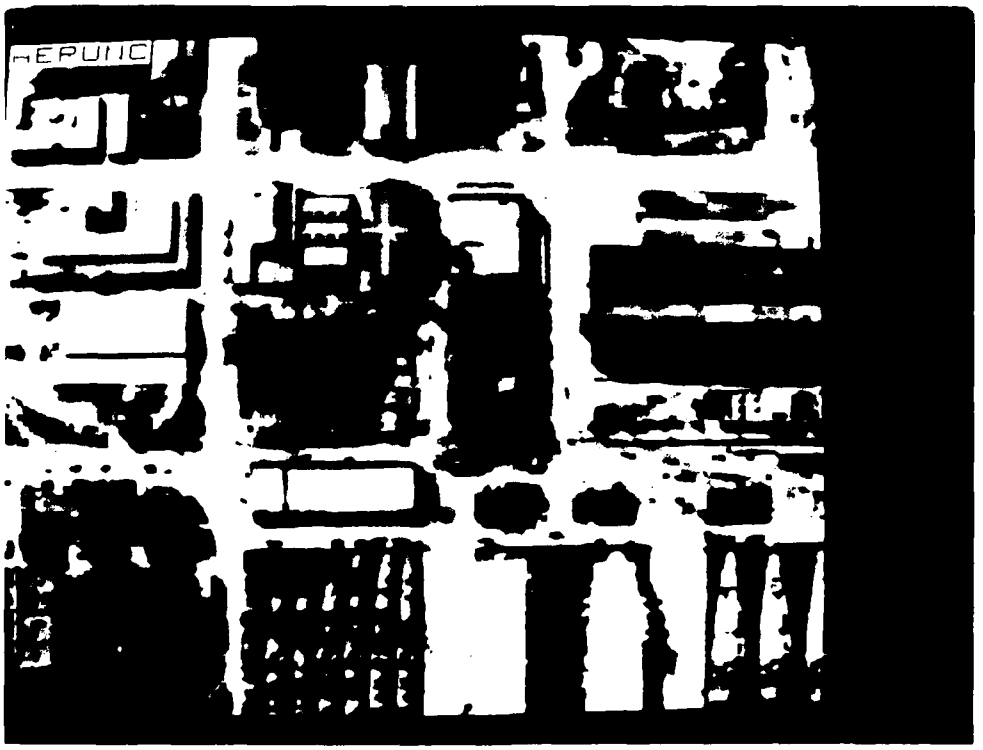


Figure 5.34 - Aerial Photo Output Image
NTV=12000, SNR=30

Figure 5.35 - Aerial Photo Output Image
NTV=6000, SNR=50

Figure 5.36 - Aerial Photo Output Image
NTV=6000, SNR=40

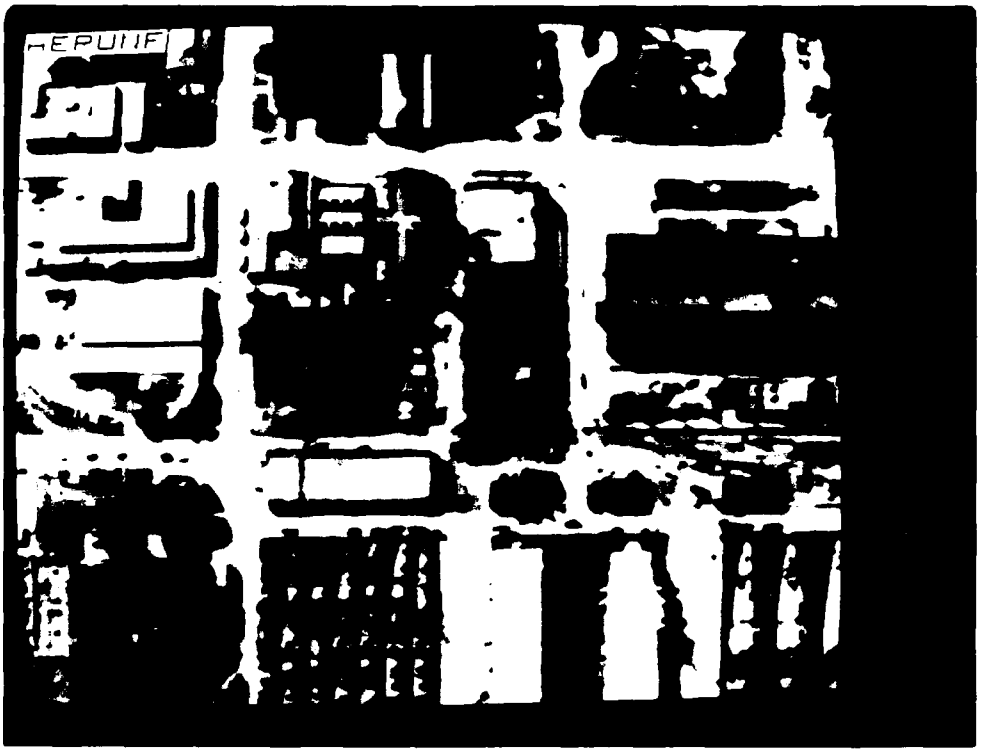


Figure 5.37 - Aerial Photo Output Image
NTV=6000, SNR=30

Figure 5.38 - Aerial Photo Output Image
NTV=3000, SNR=50

Figure 5.39 - Aerial Photo Output Image
NTV=3000, SNR=40

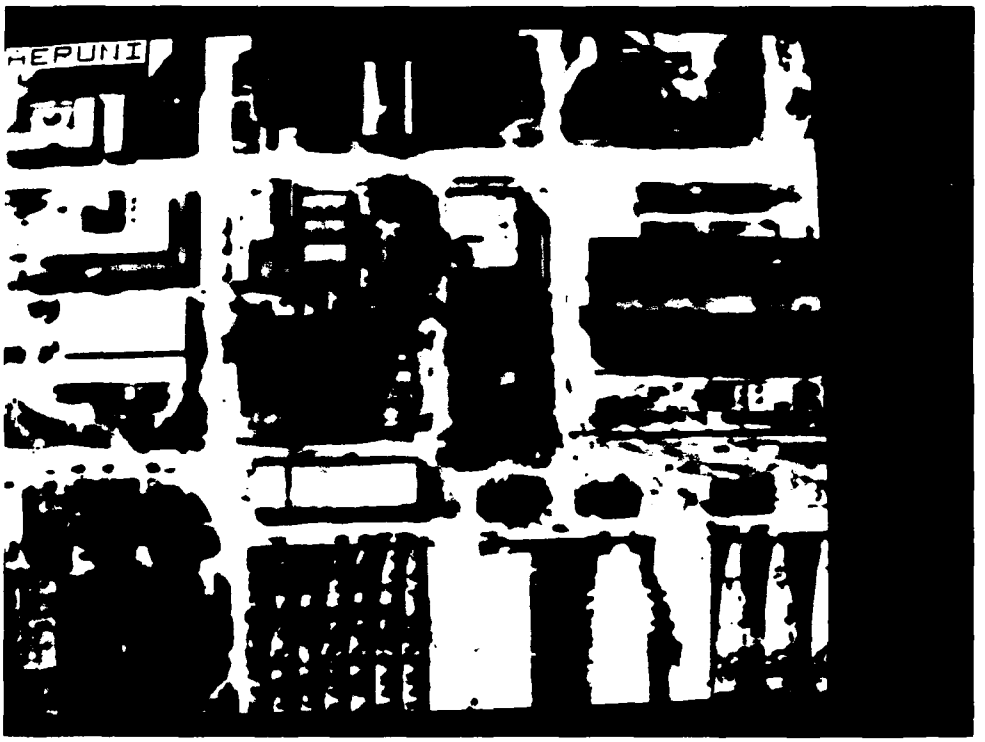


Figure 5.40 - Aerial Photo Output Image
NTV=3000, SNR=30

6.0 Conclusions and Recommendations

6.1. Conclusions

The vector quantization method selected for these simulations consists of the transmission of both scaler and vector information.

For each image block, the scaler information consists of two items: the mean value of the block pixel gray levels (analogous to brightness) and the gain (analogous to contrast), which is the dot product of the difference vector and the selected codebook vector. The difference vector is the vector derived by subtracting the block mean from the original block pixels. The vector information is the index to the selected codebook vector.

In the present system, the mean and gain are transmitted by straight PCM, instead of DPCM, at a cost of 8 bits for each, i.e. 16 bits per block for both. Since each block contained 16 pixels, the cost of transmitting the mean and gain was 1 bit per pixel. The vector information is transmitted at a cost of only a few tenths of a bit per pixel. This remarkable compression of the vector data is a result of normalizing the codebook vectors and employing the amplitude-adaptive tree search. This approach takes advantage of the fact that the vector magnitudes are usually small, and that low-magnitude vectors, which can be more coarsely quantized than high, require shorter codebook vector index word lengths.

The simulations, whose results have been presented herein, show that the present system can transmit imagery with very good

quality at an average rate of approximately 1.3 to 1.5 bits per pixel, and moderately good quality at a rate of 1.1 to 1.2 bits per pixel. Since the present system transmits the means and gains at a total cost of 1 bit per pixel, the cost of transmitting the vector data is only 0.3 to 0.5 bits per pixel for very good quality and 0.1 to 0.2 bits per pixel for moderately good quality.

It is conservatively estimated that further effort could reduce the total cost of transmitting the means and gains to approximately 0.7 bits per pixel with little performance degradation, and a goal of 0.5 bits would be realistic. With a cost of 0.5 bits per pixel for the scaler data, the total average bit rate would be 0.6 to 0.7 bits per pixel for moderately good quality and 0.8 to 1.0 bits per pixel for very good quality. These estimates are based on the present block size of 16 pixels.

The simulation results lead to the following additional conclusions:

1. The adaptive amplitude tree search and normalization of the codebook vectors yield very efficient vector data transmission by straight binary coding of the relative codebook vector indices. Table 6.1 compares the bit rates for the vector data with straight binary coding to that with theoretical minimum-entropy coding. Entropy coding would reduce the bit rate by only a few hundredths of a bit per pixel.
2. Building the training vectors from a composite image

derived from three of the four test images, sorting the training vectors into descending order of training vector magnitude (highest-contrast blocks come first) and selecting only the first 12000 sorted training vectors from a total of 65000 lead to a robust codebook with no evidence of memorization. While different images yielded different bit rates for a given image quality, there was no evidence of correlation between compression and whether or not a test image was included in the training image. The IEEE face, which was not included in the training image, had better compression than two of the three other images. Only the house and sky image was better.

3. While RMS error, or some similar objective distortion measure, is the only tractable approach to generating a near-optimal codebook, a subjective measure is the only valid gauge of how well the system performs. A case in point is the house with trees image, which has, for a 50-dB codebook with 12000 training vectors, an RMS error of 6.7 vs. 2.08 for the IEEE face, 2.29 for the house and sky and 3.43 for the aerial photograph; yet all four images have very good subjective image quality. The probable reason for this disparity is that this image is very "busy." While busy images generally have fairly large RMS errors, the distortion tends to be camouflaged.

4. Subjective distortion consists mainly of a combination of blocking and loss of detail. "Blocking" refers to false edges between adjacent blocks when the vector quantization is not fine enough to correct for large differences between adjacent block means. As the codebook signal-to-noise ratio goal was increased from 30 to 50 dB, blocking was reduced in both severity and prevalence, and detail rendition improved. With the 50-dB, 12000-training-vector codebook, blocking was virtually absent and detail rendition was excellent.

6.2. Recommendations for Further Study

6.2.1. Scaler Data Compression

Since the present system requires one bit per pixel for the scaler data, which are transmitted by PCM instead of DPCM, and only a few tenths of a bit per pixel for the vector data, effort should be directed toward compressing the scaler data.

The existing DIS DPCM system performs well for ordinary images, but not so well for mean and gain "images," each of which has one "pixel" (mean or gain value) per (16-pixel) block of the original image.

A mean image has less correlation between adjacent "pixels" than does an ordinary image; hence the existing predictor is suboptimal. Moreover, the 3-bit quantizer is probably too coarse to handle the large prediction errors. It is estimated

that with the predictor coefficients tailored to mean "images," with finer prediction error quantization than the present 3 bits, and with variable-length coding, the mean values could be transmitted in an average of 4 bits with acceptable error.

A gain image has large areas of slow gray-level variation punctuated by occasional sharp spikes. Viewed on a monitor, a gain image resembles a white-on-black outline drawing like a chalk drawing on a blackboard. The white outlines occur at edges and similar abrupt changes of gray level in the original image. A histogram of a gain image from a simulation with a 50-dB codebook generated from 12000 training vectors (best image quality) shows that with variable-length coding, 8-bit gain values could probably be transmitted error-free at an average rate of 6 bits without employing prediction. An appropriate prediction scheme, for example simple delta modulation, may lead to further compression, e.g. 4 or 5 bits, with slight error.

The combination of 4 bits for the mean and 6 bits for the gain, spread over a 16-pixel block, gives a conservative mean/gain transmission rate of 0.625 bits per original image pixel. Since the vector data compress to 0.1 to 0.5 bits per pixel, the overall transmission rate should be less than one bit per pixel with good image quality, and slightly over 1 bit per pixel (e.g. 1.1) with very good image quality. If the mean/gain rate can be reduced to 0.5 bits per pixel, then overall compression rates of considerably less than one bit per pixel can be expected.

6.2.2. Interpolative Vector Quantization

Hang and Haskell[11] describe a vector quantization method in which the difference vector to be quantized is formed by subtracting from the original block pixel values an interpolative surface instead of the block means. Instead of transmitting the block means by PCM or DPCM, the transmitter transmits a sample pixel, e.g. the upper left, of each block. The transmitter and receiver both compute, for all the pixels in any one block, estimated values which interpolate the sample values of the current and neighboring blocks. The differences between the actual pixel gray levels in the block and the estimated values are vector quantized.

Interpolative vector quantization reduces the blocking artifact, because interpolation tends to smooth the block edges prior to vector quantization. Moreover, it takes advantage of correlation over neighboring blocks, and therefore would probably reduce the estimation error that must be vector-quantized. Reduced estimation error leads to smaller residual vector magnitudes, which, with the present amplitude-adaptive normalized vector quantization system, could yield even better vector data compression than was achieved with the present system.

6.2.3. Signed Gain Values

In the present system the gains always have non-negative values. If the mismatch between a vector to be quantized and the

selected codebook vector is so bad that the dot product is negative, the gain is treated as zero, and the pixels of the reconstructed block all have the mean value. Allowing the gains to have both positive and negative values would provide for the possibility of a given codebook vector's matching, exactly or approximately, a difference vector and its negative. For example, a light-to-dark and a dark-to-light edge with the same relative position and orientation in their respective blocks would require two codebook vectors in the present system, but only one with signed gain values. (A normalized correlation coefficient of -1 is just as good as +1.) Signed gain values may therefore improve both image quality and compression.

Introducing signed gain values has the drawback of requiring one more bit of gain data per block or one less bit of gain precision. Experimentation would be required to determine whether the advantage outweighs the drawback.

6.2.4. Larger Block Size

A larger block has the potential of producing greater data compression at the expense of a larger codebook. The scaler data (e.g. mean and gain) would be spread over more pixels, but not in proportion to the number of pixels per block. If the mean values are taken over larger blocks, there would be more variation between adjacent mean values and larger residual error values to vector quantize after subtracting the mean. If interpolation is employed, the estimation error gets worse as the points between

which the pixels are interpolated become more widely spaced. More than one such point per block may need to be transmitted.

Because the maximum difference vector magnitude increases as the square root of the block size, more than 8 bits of gain precision may be required for the gain values, or the gains may have to be scaled down during transmission, with consequent loss of precision. Thus, while increased block size should reduce the bit rate of the vector data, reduction in the scaler data bit rate may be less than proportional to the block size.

Estimating codebook size as a function of block size is difficult because of vector clustering. Any size estimate that ignores clustering would be astronomical. The codebook memory requirements vary by more than the square of the block size, because the vector record length is proportional to the block size, and the number of vectors is considerably more than proportional. A rough estimate of the number of codebook vectors required for a larger block could be derived by evaluating the numbers required for, say, a 50-dB signal-to-noise ratio goal with block sizes of 4 and 9 as well as the present 16, and then extrapolating to the larger block size.

6.2.5. Codebook Generator Refinements

6.2.5.1. Faster Algorithm

With a more complex file and data structure, the codebook generator execution time could be reduced by a factor of approximately half the number of search tree levels. The faster

algorithm would point each training vector to the search tree node to which it belongs in the highest tree level which has already been optimized. This would eliminate tracing all the way from the tree root.

6.2.5.2. Signal-to-Noise Ratio Goal for Each Tree Level

Providing the option of specifying a separate signal-to-noise ratio goal for each tree level instead of one overall goal would allow the fine-tuning of a codebook to force finer quantization of low-magnitude vectors. While, from an RMS standpoint, low-magnitude vectors may be more coarsely quantized than high, small errors in smooth regions of an image are subjectively more conspicuous than in "busy" regions. The signal-to-noise ratio goal would have to be a monotone non-increasing function of tree level, because the basic system algorithm requires that the gain threshold be a monotone non-decreasing function.

6.2.6. Necessity of Separate Codebook for Each Resolution

The simulations were performed for only one image resolution value so the effects of varying several codebook parameters could be evaluated with resolution held constant. It is conjectured that a separate training image and codebook would be required for different resolutions. This could be tested quickly by simulating a few test images at different resolutions with the present 50-dB, 12000-training-vector codebook.

6.2.7. Hardware/Firmware Implementation

Once all parameters of a vector quantization system have been optimized by software simulation, a hardware/firmware implementation should be investigated. For maximum speed, as much parallel processing as possible should be built into the system. If processing is performed on-line, then speed is essential to minimize connect-time costs. If local mass storage is available, then the compressed data could be generated and stored off-line and then transmitted. The receiver could store the compressed data locally and reconstruct the image off-line. With this approach, fast processing improves total throughput, but does not reduce connect-time costs.

References

1. STANDARD GRAY SCALE IMAGES USER'S MANUAL, Delta Information Systems, Inc., NCS Contract Number DCA100-83-C-0047

2. Gersho, A., "On the Structure of Vector Quantizers," IEEE Transactions on Information Theory, V. IT-28, No. 2, March 1982, pp 157-166.

3. Linde, Y., Buzo, A., and Gray, R. M., "An Algorithm for Vector Quantizer Design," IEEE Transactions on Communications, V. COM-28, No. 1, Jan. 1980, pp. 84-95.

4. Gray, R. M. and Linde, Y., "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources," IEEE Transactions on Communication, V. COM-30, No. 2, February, 1982, pp. 381-389.

5. Huang, H-M and Woods, J. W., "Predictive Vector Quantization of Images," IEEE Transactions on Communication, V. COM-33, No. 11, November 1985, pp. 1208-1219.

6. "Component Vector Quantization," Annex 4 of CCITT Stud Group VIII, Geneva, 1-12 December 1986.

7. Helden, J. and Boekee, D. E., "Vector Quantization Using a Generalized Tree Search Algorithm," Proc. 5th Symposium on Information Theory in the Benelux, Aalten, May 1984, pp. 21-27.

8.Dutch, PTT, "A 384 Kbit/s Coding Scheme Based Upon Vector Quantization for Video Conferencing."

9.Gersho, A. and Ramamurthi, B., "Image Coding Using Vector Quantization," Proc. ICASSP, 1982, Paris.

10. Computer Simulation of Gray Scale Compression Technique for Group 4 Facsimile, Final Report to National Communications System under Defense Communications Agency Contract Number DCA100-83-C-0047, Task Order Number 84-002

11. Hsueh-Ming Hang and Barry G. Haskell, "Interpolative Vector Quantization of Color Images," IEEE Transactions on Communications, V. 36 No. 4, April, 1988, pp 465-470